Learning Collaboration Strategies for Committees of Learning Agents

Enric Plaza (enric@iiia.csic.es)
Artificial Intelligence Research Institute (IIIA)
Consejo Superior de Investigaciones Científicas (CSIC)
Campus UAB, 08193, Bellaterra, Spain

Santiago Ontañón (santi@maia.ub.es)
University of Barcelona (UB)
Gran Via 585, 08007, Barcelona, Spain

Abstract.
A main issue in cooperation in multi-agent systems is how an agent decides in which situations is better to cooperate with other agents, and with which agents does the agent cooperate. Specifically in this paper we focus on the following problem: given a multi-agent system composed of learning agents, and given that one of the agents in the system has as a goal to predict the correct solution of a given problem, the agent has to decide whether to solve the problem individually or to ask for collaboration to other agents. We will see that learning agents can collaborate forming *committees* in order to improve performance. Moreover, in this paper we will present a proactive learning approach that will allow the agents to learn when to convene a committee and with which agents to invite to join the committee. Our experiments show that learning results in smaller committees while maintaining (and sometimes improving) the problem solving accuracy than forming committees composed of all agents.

1. Introduction

A main issue in cooperation in multi-agent systems is how an agent autonomously decides in which situations is better to cooperate with other agents, and with which agents does the agent cooperate. Specifically in this paper we focus on the following scenario: assuming a multi-agent system composed of learning agents (each one in principle with a different background), and that the agents in the system have as a goal to achieve a high accuracy on predicting the correct solution of the problems they encounter, each agent has to decide when encountering a new problem whether to solve it individually or to ask for collaboration to other agents. When taking those decisions, the agent has to consider whether, for each specific problem, collaborating with other agents is likely to improve the prediction accuracy.

This generic scenario can be exemplified in the domain of marine biology where the difficult task of identifying marine sponges prompted us to address it as a multiagent system. The basic issue to be addressed

is that no biologist expert in limnology and benthology has complete knowledge of all forms and kinds of marine sponges. In practice, biologists expert in marine sponges collect specimens on their own on different parts of the world, developing a partial expertise depending on location and species. Moreover, some species change according to the location they live. Therefore, there will be no single biologist that is an expert in all kinds of sponges and all kind of oceans: they refer this fact as people having different backgrounds. When one of the biologists finds a new sponge, he needs to identify specific species of the new sponge. However, since she is not an expert on all kinds of sponges, often she will realize she has low confidence in determining the correct species of the new sponge. In this context, the expert will likely ask other biologists for counsel on the correct species of the new sponge. Notice that the biologist has to take two decisions: a) deciding when she better asks counsel to other biologists (i.e. when to collaborate with other biologists), and b) decide which of the other biologists he will ask counsel to.

In this paper we propose a multi-agent system approach to deal with such scenarios. In our approach, we will consider that each biologist has a learning agent that has access to all the marine sponges collected (and properly classified) by him. Each learning agent is able to identify a new sponge based on previous experience. However, if the agent cannot produce a prediction with high confidence, it can decide to collaborate with other agents. Thus, (analogously to the biologist behavior) the agent has to take two decisions: when to start collaboration with other agents, and with which of the other agents to collaborate. Taking these two decisions properly is crucial, since the correctness of the predictions made by the learning agent strongly depends on them.

One of our goals is to show that, through collaboration, individual learning agents and multi-agent systems can improve their performance. Both learning and collaboration are ways in which an agent can improve individual performance. In fact, there is a clear parallelism between learning a collaboration in multi-agent systems, since they are ways in which an agent can deal with its shortcomings. Let us show which are the main motivations that an agent can have to learn or to collaborate:

Motivations to learn:

1. Increase the quality of solutions (e.g. accuracy),
2. Increase efficiency,
3. Increase the range of solvable problems.

Motivations to collaborate:

1. Increase the quality of solutions (e.g. accuracy),
2. Increase efficiency,
3. Increase the range of solvable problems,
4. Have access to resources that only other agents can use.

Therefore, learning and collaboration are very related. In fact, with the exception of motivation to collaborate number 4 above, they are two extremes of a continuum of strategies to improve performance. An agent may choose to increase performance by learning, by collaborating, or by finding an intermediate point that combines learning and collaboration in order to improve performance. Specifically, we are interested in studying how an individual learning agent can improve its performance by collaborating with other agents, and how can a learning agent decide whether it is better to work individually or to cooperate with others.

Moreover, returning to the motivating example, we model the situation when a group of biologists collaborate to identify a given marine sponge as the institution we usually call *committee*. When a committee of human biologists is formed, the individual biologists have their own background and produce their individual predictions. We consider a committee goes through two main phases: discussion and deliberation. During discussion, different alternatives are presented and arguments justifying or attacking alternatives are exchanged; during deliberation, one of the alternatives is chosen by some voting system that determines the winner. Analogously, we use the notion of electronic institutions [11] for a group of learning agents collaborating to reach a join prediction, i.e. forming a *committee of agents*. In this paper we will focus only on the deliberation phase, although current work on the argumentative phase is published elsewhere [26].

Specifically, there are two core aspects we want to address in this paper, namely when a committee is needed or not, and which agents should be invited to join the committee. Notice that, in our biology scenario, a biologist often solves the sponge identification task individually, while some other times recourse to external counsel (in our model, convenes a committee) because she estimates she is not competent with respect to the problem at hand. Our approach to this issue is equipping the agents with a *competence self-model* capable of estimating if the agent is competent (or to which degree it estimates might be competent) to solve a specific problem. Moreover, we will show how this *competence self-model* can be individually learnt by every agent in the course of its regular process of solving problems and collaborating with other agents.

Concerning the second issue, notice that when a biologist decides to consult some other biologists, she does not call all the available sponge experts in the world, but just a small sample, enough to correctly identify the new sponge. For a multiagent system this means that an agent convening a committee will not simply invite always all agents in the system to join the committee. Thus, we distinguish two types of strategies for convening committees: fixed committees and *dynamic committees*. An agent convenes fixed committees when the agents invited to join the committee are always the same regardless of the problem to be solved (an example of this strategy is the basic one of always convening *all* the available agents to a committee). Moreover, when an agent convenes a dynamic committee it has to select which agents to invite in function of the problem to be solved. In this paper we propose to equip each individual agent with *competence models* of the other agents; these competence assess the confidence of the convener agent in that some other agent is competent to solve the problem at hand. Moreover, we will show how these *competence models* can be individually learnt by every agent in the course of its regular process of solving problems and collaborating with other agents (Section 3.1).

1.1. Committees and Machine Learning

Committees allow us to study the application of machine learning techniques to multi-agent systems, and the relation between collaboration and learning. From a machine learning perspective, a committee may be considered an ensemble of agents, where each agent plays the role of a predictor (trying to predict the correct solution for a given problem). Ensembles of predictors are expected to have a higher performance than individual predictors because of the *ensemble effect* [27]. The ensemble effect is well known in Machine Learning, and states that, given that some preconditions are met, the combination of predictions made by several individual predictors is likely to be more accurate than the prediction made by the individual predictors. The preconditions of the ensemble effect are simple: each individual predictor must be minimally competent (i.e. have an error rate lower than 0.5) and the ensemble must be diverse (i.e. the error correlation between the predictions of the individual classifiers must be low).

In previous work [28], we have shown that committees can also benefit from the ensemble effect, as ensembles of predictors do. Thus, by properly defining strategies to convene committees, agents can convene committees that allow them to achieve higher performance than working individually. However, committees are not the same as ensembles, in other words our goal is not to present new ensemble learning methods.

The fundamental differences between committees and ensembles are, for instance, that autonomy and privacy are not an issue in ensemble learning, but they ar essential in multi-agent systems. Moreover, in an ensemble, the ensemble learning algorithm is *centralized* and *creates* the individual predictors in such a way that the ensemble works; however, in a committee, agents are not created by a centralized process, agents are in principle created or maintained by different organizations in different places; therefore, in our multiagent framework an agent has to convene a committee that achieves the maximum performance by collaborating with the existing agents, and having no control or access to the data they have stored locally. These hypotheses of *decentralized control* and *distributed data* that our framework espouses are not satisfied by ensemble learning methods that assume centralized control and access to data. Therefore ensemble learning methods are not directly applicable to committees, although committees can use the core ideas of the "ensemble effect" to improve their performance [28].

1.2. An Approach to Learning to Cooperate

The problem of convening dynamic committees is presented in this paper inside a framework called *Multi-agent Case Based Reasoning Systems* ($\mathcal{M}$AC)[28]. A $\mathcal{M}$AC system is composed by a set of CBR agents, where a CBR agent is an agent that uses Case Based Reasoning (CBR) [1] to solve problems and learn from those problems. The open and dynamic nature of multi-agent systems fits with open and dynamic nature of *lazy learning* [2] used in CBR. This framework is quite general and has been used to study different aspects concerning learning in multiagent systems [28, 21, 20, 23, 29, 22, 25].

In this paper, however, we focus on presenting *collaboration strategies* that the agents in a $\mathcal{M}$AC system can use to convene committees in the phase of join deliberation (and therefore excluding the argumentation phase). For this reason, we will use a $\mathcal{M}$AC system where agents learn to perform a classification tasks without lose of generality. In Machine Learning, a classification task is one where a learning system predicts one item among a set alternatives (usually called classes). Since we are focusing on the deliberation phase of committees, where one of the presented alternatives has to be selected, this task is, from the point of view of the learning agent, a classification task. Notice our approach is general in the sense that the alternatives under deliberation can be internally complex (e.g. a committee can deliberate on alternative plans of action), but in the deliberation phase the individual agents have just to predict (and learn to predict) the better alternative.

Specifically, we will present a basic strategy called the *Committee Collaboration Strategy* (CCS) that always convenes a committee using all the available agents in the system. CCS is a strategy for *fixed committees* and is used for comparison purposes since using all agents in a committee will (in principle) lead to more accurate predictions. After that, we will present another strategy called *Proactive Bounded Counsel Collaboration Strategy* (PB-CCS), that tries to achieve accurate predictions, but only convening committees when required. Thus, PB-CCS is our proposal to address the problem of deciding when to collaborate, and with which agents to collaborate. We will present specific *decision policies* that agents may use to decide when to solve problems individually and when to convene committees, and to select which agents to convene to a committee. Moreover, the key claim of this work is that agents can learn to make those decisions (when to collaborate and with which agents to collaborate). To support this claim, we present a proactive learning approach that gives the agents the ability to learn how to take those decisions.

Since those decision policies are based on what we call *competence models*, we will two approaches: one where the competence models are predetermined (and manually build by the agent designers) and another one where those competence models are individually learnt by every agent in the systems using PB-CCS. We also present experiments to compare the performance fixed vs. dynamic committees, and that of learning competence models vs. predetermined competence models.

The structure of the paper is as follows. First Section 2 presents the multi-agent framework in which we have performed our experiments, and formally define the notion of committee. Moreover, Section 2 presents the *Committee Collaboration Strategy*. After that, Section 3 introduces the notion of *dynamic committees*, and the *Proactive Bounded Counsel Collaboration Strategy*, that will be presented a a dynamic committee collaboration strategy. Then, Section 4 presents a proactive learning technique with which agents will be able to learn a decision policy used to convene dynamic committees. Section 5 formally presents the *Bounded Counsel Collaboration Strategy* for comparison purposes. Finally, Section 6 presents an empirical evaluations of all the collaboration strategies in several scenarios. The paper closes with related work and conclusions sections.

2. A Multi-agent CBR Approach

In this paper we focus on agents that use Case Based Reasoning (CBR) to solve problems. CBR techniques suit perfectly into multi-agent sys-

tems and give the agents the capability of autonomously learn from experience by retaining new cases (problems with known solution). Therefore, we will focus on *Multi-agent CBR Systems* ($\mathcal{M}$AC).

DEFINITION 2.1. *A* Multi-Agent Case Based Reasoning System *($\mathcal{M}$AC)* $\mathcal{M} = \{(A_1, C_1), ..., (A_n, C_n)\}$ *is a multi-agent system composed of a set of CBR agents* $\mathcal{A} = \{A_i, ..., A_n\}$ *where each agent* $A_i \in \mathcal{A}$ *possesses an individual case base* C_i.

Each individual agent A_i in a $\mathcal{M}$AC is completely autonomous and has access only to its individual and private case base C_i. A case base $C_i = \{c_1, ..., c_m\}$ is a collection of cases. Each agent has (in general) its own CBR method(s) to solve problems using the cases stored in its individual case base. Agents in a $\mathcal{M}$AC system are able to individually solve problems, but the can also collaborate with other agents to solve problem in a collaborative way.

In this paper, we will focus on classification tasks, where the solution of a problem is achieved by selecting a solution class from an enumerated set of solution classes. We have taken this decision because aggregation of predictions in classification domains can be easily achieved using any voting system. Other domains would require more complex aggregation mechanisms, and it is out of the scope of this paper to define complex aggregation methods for predictions in complex domains such as planning or configuration.

In the following we will note the set of all the solution classes by $\mathcal{S} = \{S_1, ..., S_K\}$. Moreover, we will note the problem space by $\mathcal{P}$, that contains all the problems that can be described in a particular application domain. Therefore, a case can be defined as:

DEFINITION 2.2. *A* case $c = \langle P, S \rangle$ *is a tuple containing a case description* $P \in \mathcal{P}$ *and a solution class* $S \in \mathcal{S}$.

Notice that case descriptions are defined over the problem space $\mathcal{P}$. In the following, we will use the terms *problem* and *case description* indistinctly. Therefore, we can say that a case consists of a case description plus a solution class, or that a case is a problem/solution pair. Moreover, we will use the dot notation to refer to elements inside a tuple. e.g., to refer to the solution class of a case c, we will write $c.S$. Moreover, we will also use the dot notation with sets, i.e. if C is a set of problems, $C.P$ refers to the set of problems contained in the cases in C, i.e. $C.P = \{c.P | c \in C\}$.

Moreover, in our framework, all the interaction among agents is performed by means of *collaboration strategies*.

DEFINITION 2.3. *A* collaboration strategy $\langle I, D_1, ..., D_m \rangle$ *defines the way in which a group of agents inside a* $\mathcal{M}$AC *collaborate in order to achieve a common goal and is composed of two parts: an interaction protocol* I*, and a set of individual decision policies* $\{D_1, ..., D_m\}$*.*

The *interaction protocol* of a collaboration strategy defines a set of interaction states, a set of agent roles, and the set of actions that each agent can perform in each interaction state. The agents use their individual *decision policies* to decide which action to perform, from the set of possible actions, in each interaction state. Each agent is free to use its own decision policies. Moreover, we have used the ISLANDER formalism [10] to specify the interaction protocols in our framework.

2.1. A CBR view of $\mathcal{M}$AC systems

In this section we will try to present a CBR view of the collaboration strategies presented in this paper. For that purpose, let us briefly explain how CBR works.

The CBR problem solving cycle consists of four processes: *Retrieve*, *Reuse*, *Revise*, and *Retain* [1]. During the Retrieve process, a CBR system searches its case base for cases that can be used to solve the problem at hand (relevant cases); during the Reuse process, the solution of the cases retrieved during the Retrieve process is used to solve the problem at hand. Thus, after the Retrieve and Reuse processes, the CBR system has already solved the problem. After that, in the Revise process, the solution provided by the system is revised by an expert or by a causal model to ensure that the solution is correct, and a new case is constructed using the problem and the revised solution. Finally the Retain process decides whether the new case should be incorporated into the case base for future use.

When a group of agents in a $\mathcal{M}$AC system collaborate to solve problems, they also follow the CBR cycle. Specifically, our proposal is that each individual agent should perform the Retrieve process individually since case bases are private, and no agent should have access to the case base of another agent. Collaboration will take place during the Reuse process, where agents will collaborate to decide an overall solution for the problem at hand by aggregating their individual predictions. Revise and Retain processes fall out of the scope of this work (see [24] for work on collaborative retention).

Thus, all the collaboration strategies presented in this paper have to be seen as strategies that take place during the Reuse CBR cycle.

2.2. Committees of CBR agents

This section presents the notion of *Committees of agents* that allows a group of agents to benefit from the ensemble effect by collaborating when solving problems.

DEFINITION 2.4. *A* Committee *is a group of agents that join together to predict the solution of a problem P. Each agent individually predicts the solution of P and then all the individual predictions are aggregated by means of a voting process.*

The only requirement on the CBR method that an agent in a committee uses is that after solving a problem P, and agent A_i must be able to build a *Solution Endorsement Record*. A *Solution Endorsement Record* (SER) is a tuple $\mathbf{R} = \langle S, E,\ P, A\rangle$ where A is an agent that has found E (where $E > 0$ is an integer) cases endorsing the solution S as the correct solution for the problem P. Intuitively, a SER is a record that stores the result of individually performing the Retrieve CBR process. If the CBR method of an agent can return more than one possible solution class, then a different SER will be built for each solution.

When a committee of agents solves a problem, the sequence of operation is the following one: first of all, an agent receives a problem to be solved and convenes a committee to solve the problem; the problem is sent to all the agents in the committee and every agent preforms the Retrieve process individually; after that, instead of performing Reuse individually, each agent reify the evidence gathered during the Retrieve process about the likely solution(s) of the problem in the form of a collection of SERs. The Reuse process is performed in a collaborative way by aggregating all the SERs to obtain a global prediction for the problem. In our experiments, agents use a voting process (see Section 2.4) to aggregate predictions.

Moreover, there are many different strategies to convene committees. In the remainder of this paper we will present several collaboration strategies that convene different types of committees.

2.3. Committee Collaboration Strategy

This section presents the *Committees Collaboration Strategy* (CCS). Specifically, the Committee Collaboration Strategy is composed by an interaction protocol and an individual decision policy:

DEFINITION 2.5. *The* Committee Collaboration Strategy *(CCS) is a collaboration strategy $\langle I_C, D_V\rangle$, where I_C is the CCS interaction*

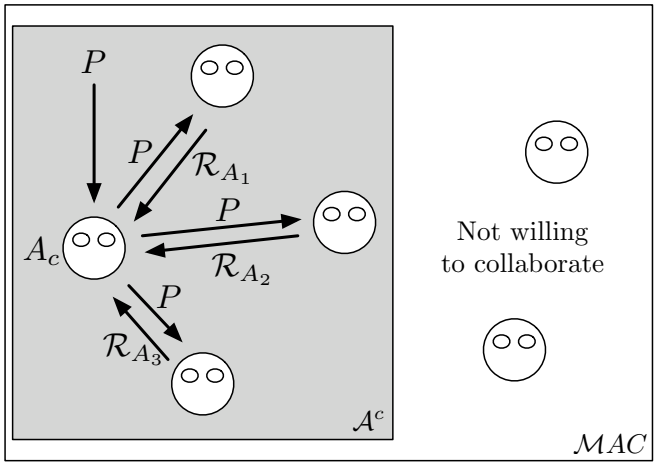


Figure 1. Illustration of a $\mathcal{MAC}$ system where an agent A_c is using CCS in order to convene a committee to solve a problem.

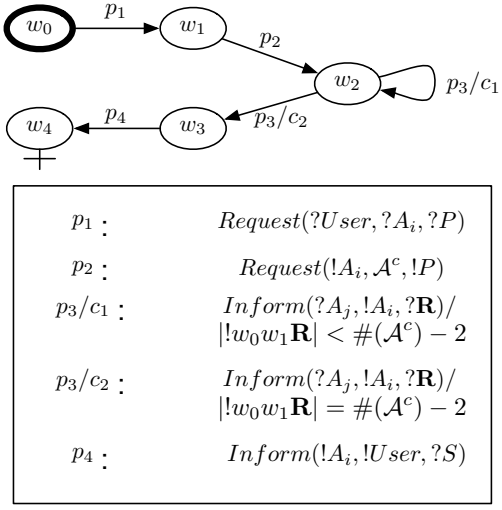


Figure 2. Interaction protocol for the *Committee* collaboration strategy.

protocol shown in Figure 2 and D_V is a decision policy based on any voting system that can be used to aggregate the evidence gathered by the individual agents into a global prediction (specifically, we will use a voting system called BWAV, presented in Section 2.4).

The interaction protocol I_C is described in Figure 2 using the ISLANDER [10] formalism and applies to a set of agents $\mathcal{A}^c$ that have agreed to join a committee. The protocol consists of five states and w_0

is the initial state. When a user requests an agent A_i to solve a problem P the protocol moves to state w_1. Then, A_i broadcasts the problem P to all the other agents in the system and the protocol moves to state w_2. Then, A_i waits for the SERs coming from the rest of agents while building its own SERs; each agent sends its SERs to A_i in the message p_3. When the SERs from the last agent are received the protocol moves to w_3. In w_3, A_i will apply the voting system defined in the individual decision policy D_V (with all the SERs received from other agents and the SERs built by itself) to aggregate a global prediction. Finally, the aggregate prediction S will be sent to the user in message p_4 and the protocol will move to the final state w_4.

Notice that not all the agents in the $\mathcal{MAC}$ system may be willing to collaborate using CCS. Therefore, the set $\mathcal{A}^c$ contains only those agents that are willing to collaborate, as shown in Figure 1.

Since all the agents in a $\mathcal{MAC}$ system are autonomous CBR agents, they will not have the same problem solving experience. Therefore, the cases in their case bases will not be the same. For that reason, not all the agents will be able to solve exactly the same problems, and there will be some problems that some agents fail to solve correctly but that some other agents will be able to solve. In other words, the individual agent's errors are uncorrelated. Thus, using the committee collaboration policy an agent can increase its problem solving accuracy because it satisfies the preconditions of the *ensemble effect*.

2.4. Bounded Weighted Approval Voting

Agents can use any voting system to aggregate their predictions. However, in this section we are going to present a voting system called BWAV specifically designed for committees of CBR agents. As we will see in Section 4, agents will perform a learning process that will use as input the information provided by the votes of the games. Thus, the more informative are the votes, the better the agents will be able to learn. For that reason, BWAV is more adequate than standard *majority voting* (where each agent will simple vote for a single solution). Moreover, more informative voting systems could be defined, however BWAV provides enough information for the learning process to perform well.

The principle behind the voting system is that the agents vote for solution classes depending on the number of cases they found endorsing those classes. Specifically, each agent has one vote that can be for a unique solution class or fractionally assigned to a number of classes depending on the number of endorsing cases.

Let $\mathcal{R}^c = \{\mathbf{R}_1, ..., \mathbf{R}_m\}$ be the set of SERs built by the n agents in $\mathcal{A}^c$ to solve a problem P. Notice that each agent is allowed to submit one or more SERs. In fact, an agent will submit as many SERs as different solution classes are present in the retrieved cases to solve P. Let $\mathcal{R}_{A_i} = \{\mathbf{R} \in \mathcal{R}^c | \mathbf{R}.A = A_i\}$ be the subset of SERs of $\mathcal{R}$ created by the agent A_i to solve problem P. The vote of an agent $A_i \in \mathcal{A}^c$ for a solution class $S_k \in \mathcal{S}$ is the following:

$$Vote(S_k, P, A_i) = \begin{cases} \frac{\mathbf{R}.E}{c+N} & \text{If } \exists \mathbf{R} \in \mathcal{R}_{A_i} | \mathbf{R}.S = S_k, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

where c is a normalization constant that in our experiments is set to 1 and $N = \sum_{\mathbf{R} \in \mathcal{R}_{A_i}} \mathbf{R}.E$ is the total number of cases retrieved by A_i. Notice that if an agent A_i has not created a SER for a solution class S_k, then the vote of A_i for S_k will be 0. However, if A_i has created a SER for S_k, then the vote is proportional to the number of cases found endorsing the class S_k, i.e. $\mathbf{R}.E$.

To understand the effect of the constant c we can rewrite the first case of Equation 1 as follows (assume that $\mathbf{R}$ is the SER built by A_i for the solution class S_k):

$$Vote(S_k, P, A_i) = \frac{\mathbf{R}.E}{N} \times \frac{N}{c+N}$$

Since N is the total number of cases retrieved by A_i, the first fraction represents the ratio of the retrieved cases endorsing solution S_k with respect to N (the total number of cases retrieved by A_i). The second fraction favors the agent that has retrieved more cases, i.e. if A_i has only retrieved one case, and it is endorsing S_k, then the vote of A_i for S_k will be $Vote(S_k, P, A_i) = \frac{1}{1+1} = 0.5$; moreover, if the number of retrieved cases is 3 (and all of them endorsing S_k), then the vote is $Vote(S_k, P, A_i) = \frac{3}{1+3} = 0.75$. Notice that the sum of fractional votes casted by an agent is upper bounded by 1, but in fact it is always less than 1 and, the more cases retrieved, the closer to 1. Finally, notice that if $c = 0$ the sum of votes is always 1.

We can aggregate the votes of all the agents in $\mathcal{A}^c$ for one class by computing the ballot for that class:

$$Ballot(S_k, P, \mathcal{A}^c) = \sum_{A_i \in \mathcal{A}^c} Vote(S_k, P, A_i)$$

and therefore, the winning solution class is the class with more votes in total:

$$Sol(\mathcal{S}, P, \mathcal{A}^c) = \arg\max_{S_k \in \mathcal{S}} Ballot(S_k, P, \mathcal{A}^c) \quad (2)$$

We call this voting system *Bounded-Weighted Approval Voting* (BWAV), and it can be seen as a variation of *Approval Voting* [3]. The main differences between approval voting and BWAV are that in BWAV agents can give a weight to each one of its votes and that the sum of the votes of an agent in BWAV is always smaller than 1. In *Approval Voting* each agent votes for all the candidates they consider as an acceptable outcome without giving weights to the accepted options.

3. Dynamic Committees

The Committee Collaboration Strategy (CCS) can effectively improve the problem solving performance of the agents in a $\mathcal{MAC}$ system with respect to agents solving problems individually (as we will show in the experimental results section). However, when an agent uses CCS, no policy is used to select which agents are invited to join the committee and *all* the agents in a $\mathcal{MAC}$ system are invited each time that an agent wants to use CCS. Moreover, it is not obvious that forming a committee with all the available agents is the best option for all the problems: possibly smaller committees have an accuracy comparable (or indistinguishable) to that of the complete committee. Furthermore, possibly some problems could be confidently solved by one agent while others could need a large committee to be solved with confidence. Finally, in some domains (recall the marine sponge example presented in Section 1) it is absolutely nonsense to convene all the agents.

In this paper we will study different collaboration strategies that do not invite always all the agents to join the committee. The goal of these strategies is to study whether it is possible to achieve similar accuracies than the Committee Collaboration Strategy without convening always the complete committee. We are interested in studying whether it is possible to provide agents with strategies that convene large committees only when the application domain requires it, and convene smaller ones when there is no need for large ones. Specifically, we will focus on solving two main problems:

1. Deciding when an individual agent can solve a problem individually and when it is needed to convene a committee.

2. Deciding which agents should be invited to join the committee.

A collaboration strategy that convenes a different committee in function of the current problem is called a *Dynamic Committee* collaboration strategy. Moreover, as we have stated in Section 1, agents require

competence models in order to decide when to convene a committee and which agents to invite.

3.1. Competence Models

All the strategies presented in this paper use *competence models* in order to decide which agents will form a committee.

DEFINITION 3.1. *A* competence model $M_A(P) \rightarrow [0,1]$ *is a function that estimates the confidence on the prediction of an agent (or set of agents) for a specific problem P, i.e. estimates the likelihood that the prediction is correct.*

Competence models can be acquired by two different ways: a) directly specified by a human user, b) automatically learned from experience by the agents. In this paper we will present a learning technique to allow agents to learn their own competence models. Moreover, in the experimental results section we will compare the learned competence models against fixed and handcrafted competence models.

Competence models will be used for two purposes: a) to assess the confidence of a given committee and decide whether inviting more agents to join the committee could improve performance, and b) to assess the confidence of agents that have not yet joined in order to decide which of them should be invited to join the committee. A central issue for these decisions is to assess the confidence of a set of collaborating agents $\mathcal{A}^c$, including the special case of a committee composed of a single agent (the convener agent), that corresponds to assessing the confidence of a single agent individually solving a problem.

Therefore, a competence model must assess the competence of an agent or group of agents given a voting situation, i.e. a situation in which committee has been convened and the convener agent is ready to apply a voting system to obtain a final prediction for the problem. Notice that the collection of SERs $\mathcal{R}_{\mathcal{A}^c}$ casted by the agent members of a committee $\mathcal{A}^c$ completely characterizes a voting situation (since from $\mathcal{R}_{\mathcal{A}^c}$ we can obtain which agents are members of the committee and which have been their votes).

DEFINITION 3.2. *A* voting situation $\mathcal{R}_{\mathcal{A}^c}$ *is a set of SERs for a problem P sent by a committee of agents $\mathcal{A}^c$ to the convener agent (including the SERs of the convener agent A_c).*

For each voting situation we can define the *candidate solution* of a voting situation as the solution that the committee will predict if no more agents join the committee: $S^c = Sol(\mathcal{S}, P, \mathcal{R}_{\mathcal{A}^c})$. Moreover, we

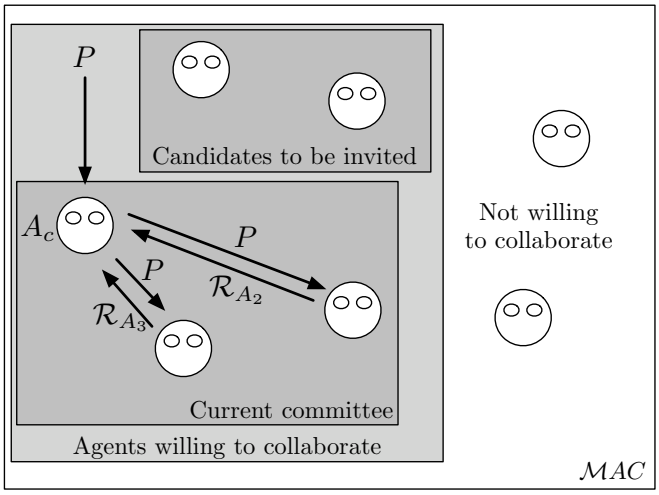


Figure 3. Illustration of PB-CCS where 3 agents have already been invited to join the committee, forming a committee of 4 agents.

can also define the individual candidate solution of an agent A_i in a committee as the solution that A_i individually predicts for the problem: $S^c_{A_i} = Sol(\mathcal{S}, P, \mathcal{R}_{A_i})$.

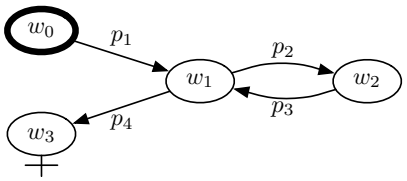
Previously, we have given a general definition for a competence model (Definition 3.1). In our approach, a competence model specifically takes as input a voting situation $\mathcal{R}_{\mathcal{A}^c}$ and outputs a confidence value in the interval $[0, 1]$. The output represents the confidence that the candidate solution of the voting situation is correct. If the competence model is modelling the competence of a single agent A_i, then the output represents the confidence that the individual candidate solution of A_i is correct.

In Section 4 we will present a proactive technique to learn competence models, and in Section 5 we will present an example of a predefined competence model.

3.2. Proactive Bounded Counsel Collaboration Strategy

The *Proactive Bounded Counsel Collaboration Strategy* (PB-CCS) is designed to study if the decisions that have to be taken to convene dynamic committees can be learnt. Specifically, agents using PB-CCS will engage in a proactive process to acquire the information they need in order to learn a *decision policy* that allows them to decide *when* and *which* agents will be invited to join each committee.

Before explaining the proactive learning process, we will first introduce how a dynamic committee is convened. For this purpose, we



p_1 :	$Request(?User, ?A_i, ?P)$
p_2 :	$Request(!A_i, ?A_j, !P)$
p_3 :	$Inform(!A_j, !A_i, ?\mathbf{R})$
p_4 :	$Inform(!A_i, !User, ?S)$

Figure 4. Interaction protocol for the *Proactive Bounded Counsel* collaboration strategy.

propose an iterative approach to determine the committee needed to solve a problem. The iterative approach works as follows: In the first round, only the convener agent individually predicts the solution of the problem. Then, a competence model is used to determine whether there is enough confidence on the individually predicted solution. It there is enough confidence, then no committee is convened, and the prediction made by the convener agent is considered the final solution. However, if there is not enough confidence, then a committee is convened in the subsequent rounds: a new agent A_j is invited to join the committee in the second round; the committee of two agents solve the problem and a competence model is used again to determine whether there is enough confidence on the solution predicted by that committee. If there is not enough confidence a new agent is invited in a third round, and so on. When the competence model estimates that a prediction has enough confidence, the process terminates and the solution predicted is returned. Figure 3 illustrates this process: from all the agents in the $\mathcal{M}$AC system that have agreed to collaborate, some of them have already joined the committee, and some of them are candidates to be invited if the confidence in the solution predicted by the current committee is not high enough. Moreover, notice that some agents in the $\mathcal{M}$AC system may be unwilling to participate in PB-CCS(since they are autonomous and may have their own reasons for not collaborating, such as not having enough free computational resources to accomplish the task, or any other reason), thus are not candidates to be invited to join the committee.

Notice that this iterative process does not guarantee to find the optimal committee, where we consider the optimal committee to be the smaller committee with the maximum confidence (predicted using a competence model). However, computing the optimal committee may be prohibitive, since there are an exponential number of possible committees.

In order to use the iterative approach to form dynamic committees, the convener agent needs two individual decision policies (in addition to the voting system), namely a *Halting* decision policy and an *Agent Selection* decision policy.

DEFINITION 3.3. *The* Proactive Bounded Counsel Collaboration Strategy *(PB-CCS) is defined as a collaboration strategy* $\langle I_B, D_H, D_{AS}, D_V \rangle$*, consisting of an interaction protocol* I_B*, shown in Figure 4,* D_H *is the Proactive Bounded Counsel* Halting *decision policy,* D_{AS} *is the Proactive Bounded Counsel* Agent Selection *decision policy, and* D_V *is the voting decision policy based on BWAV (see Section 2.4).*

PB-CCS is an iterative collaboration strategy consisting in a series of rounds. We will use t to note the current round of the protocol; thus, $\mathcal{A}_t^c$ will be the subset of agents of $\mathcal{A}$ that have joined the committee at round t and $\mathcal{A}_t^r$ the subset of agents of $\mathcal{A}$ that have not yet been invited to join the committee at round t. Finally, we will note $\mathcal{R}_{\mathcal{A}_t^c}$ the set of all the SERs submitted to the convener agent by all the agents in $\mathcal{A}_t^c$ (included the SERs built by the convener agent A_c itself), i.e. $\mathcal{R}_{\mathcal{A}_t^c}$ represents the voting situation at round t.

Figure 4 shows the formal specification of the I_B interaction protocol. The protocol consists of 4 states: w_0 is the initial state, and, when a user requests an agent A_i to solve a problem P, the protocol moves to state w_1. The first time the protocol is in state w_1 the convener agent uses the D_H decision policy to decide whether to convene a committee not. If a committee will be convened, then the D_{AS} decision policy is used to choose an agent A_j, and message p_2 is sent to A_j containing the problem P. After that and the protocol moves to state w_2. A_i remains in state w_2 until A_j sends back message p_3 containing its own prediction for the problem P, and the protocol moves back to state w_1. In state w_1 the convener agent assesses the confidence of the current prediction and uses the D_H decision policy to decide whether another agent has to be invited to join the committee or not. If A_i decides to invite more agents, then message p_2 will be send to another agent (chosen using the D_{AS} decision policy), repeating the process of inviting a new agent; if A_i decides that no more agents need to be invited to join the committee the voting system specified in D_V will be used to aggregate a global

prediction S. Finally, A_i will send the global prediction to the user with message p_4, and the protocol will move to the final state w_3.

3.3. Proactive Bounded Counsel Decision Policies

Both D_H and D_{AS} decision policies use competence models. Thus, let us introduce the competence models used by the two decision policies before explaining them in detail. Consider an agent A_i member of a $\mathcal{MAC}$ system composed of n agents, $\mathcal{A} = \{A_1, ..., A_n\}$. In order to use PB-CCS, A_i needs to learn several competence models, namely $\mathcal{M}_{A_i} = \{M_c, M_{A_1}, ..., M_{A_{i-1}}, M_{A_{i+1}}, ..., M_{A_n}\}$, where M_c is a *Committee-Competence Model* and M_{A_j} are *Agent-Competence Models*.

A *Committee-Competence Model* M_c is a competence model that assesses the confidence in the prediction of a committee $\mathcal{A}^c$ in a given voting situation $\mathcal{R}$. Thus, M_c is used to decide whether the current committee $\mathcal{A}^c_t$ is competent enough to solve the problem P or it is better to invite more agents to join the committee.

An *Agent-Competence Model* M_{A_j} is a competence model that assesses the confidence in the prediction made by an agent A_j in a given voting situation $\mathcal{R}$. M_{A_j} is useful for the convener agent to select which agent A_j is the best candidate to be invited to join the committee by selecting the agent A_j for which its competence model predicts the highest confidence (i.e. the agent with the highest likelihood that its prediction is correct) given the current voting situation $\mathcal{R}$.

Notice that the convener agent A_i does not specifically require a competence model of itself since the competence model M_c can be used to assess its own competence. For that purpose, the convener agent A_i uses M_c to assess the confidence of a committee consisting of only one agent, itself. Thus, the Committee-Competence Model M_c can be used both to assess the individual confidence of the convener agent and to assess the confidence of a committee of agents (where A_i is the convener).

Using those competence models, we can define the Proactive Bounded Counsel Halting decision policy D_H as a boolean decision policy that decides whether the convener agent can stop inviting agents to the committee at a round t; i.e. if $D_H(\mathcal{R}_{\mathcal{A}^c_t}) = true$, no more agents will be invited to join the committee.

$$D_H(\mathcal{R}_{A_i}) = \Big(M_c(\mathcal{R}_{\mathcal{A}^c_t}) \geq \eta_1\Big) \vee \Big(max_{A_j \in \mathcal{A}^r_t}(M_{A_j}(\mathcal{R}_{\mathcal{A}^c_t})) < \eta_2\Big)$$

where η_1 and η_2 are threshold parameters. The rationale of this policy is the following: if the confidence in the solution predicted by the current committee is high enough ($M_c(\mathcal{R}_{\mathcal{A}^c_t}) \geq \eta_1$) there is no need to invite

more agents since the current prediction has a very high confidence. Moreover, if the confidence on an agent $A_j \in \mathcal{A}^r$ that is not in the committee is very low ($M_{A_j}(\mathcal{R}_{\mathcal{A}_t^c}) < \eta_2$) inviting A_j to join the committee is not advisable (since the prediction of that agent will very likely be incorrect and would increase the chances that the committee prediction is incorrect). Therefore, if the maximum confidence of every agent in $\mathcal{A}_t^r$ is very low, i.e. $max_{A_j \in \mathcal{A}_t^r}(M_{A_j}(\mathcal{R}_{\mathcal{A}_t^c})) < \eta_2$, inviting any of these agents to join the committee is not advisable. This follows from one the preconditions of the ensemble effect (see Section 1), that state that the individual members of an ensemble have to be minimally competent.

The two threshold parameters η_1 and η_2 have the following interpretation: η_1 represents the minimum confidence required for the committee's prediction (candidate solution) of the current voting situation; η_2 represents the minimum confidence required in the prediction of an individual agent to allow that agent to join the committee.

Notice that by varying η_1 and η_2, the behavior of PB-CCS can be changed. Assuming that by adding more agents to the committee the confidence of the predicted solution will tend to increase, if we set a high value for η_1, the convener agent will tend to convene larger committees; and if we set a low value for η_1, the convener agent will stop inviting agents earlier, since a lower confidence will be considered adequate enough. Moreover, by setting a high value for η_2, the convener agent will be very selective with the agents allowed to join the committee (since only those agents with a confidence higher than η_2 will be allowed to join). On the other hand, a low value of η_2 will make the convener agent very permissive, and any agent could potentially be invited to join the committee.

In fact, if $\eta_1 = 0.0$, an agent will always solve problems individually, and if the parameters are set to $\eta_1 = 1.0$ and $\eta_2 = 0.0$ the resulting collaboration strategy will always convene all the available agents in the $\mathcal{MAC}$ system, and therefore achieve the same results than the Committee Collaboration Strategy. Furthermore, by increasing η_2 (leaving $\eta_1 = 1.0$) we can obtain a collaboration strategy that invites all the agents to join the committee except those that have a confidence level lower than η_2. Therefore, η_1 and η_2 allow us to define a range of different strategies to build committees.

The second decision policy is the Proactive Bounded Counsel Agent Selection decision policy D_{AS}, that is defined as a function that takes as input a voting situation $\mathcal{R}_{A_i}$ and a set of candidate agents to be invited to the committee and returns the name of the agent that has the highest confidence on finding the correct solution for a given problem:

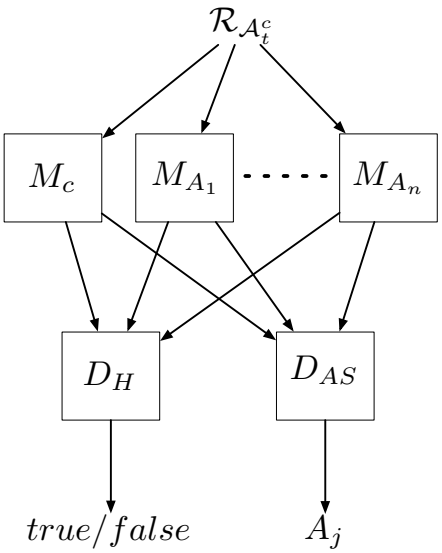


Figure 5. Relation among the competence models and the Proactive Bounded Counsel decision policies.

$$D_{AS}(\mathcal{R}_{A_i}, \mathcal{A}_t^r) = argmax_{A \in \mathcal{A}_t^r}(M_A(\mathcal{R}_{\mathcal{A}_t^c}))$$

That is to say, D_{AS} selects to invite the agent $A_j \in \mathcal{A}_t^r$ that has the highest confidence $M_{A_j}(\mathcal{R}_{\mathcal{A}_t^c})$ on predicting the correct solution.

Figure 5 shows the relations among the competence models and the decision policies in PB-CCS. The figure shows that at each round, the current voting situation, $\mathcal{R}_{\mathcal{A}_t^c}$, is the input to the competence models. Then, the output of the competence models are used as the inputs of the decision policies.

Moreover, notice that only the convener agent needs to use competence models. However, any agent that wants to use PB-CCS to convene dynamic committees has to maintain its own competence models.

4. Proactive Learning

This section presents a proactive learning technique with which an agent A_i in a $\mathcal{MAC}$ system can learn its competence models $\mathcal{M}_{A_i}$ to be used in PB-CCS. In order to learn these competence models, agents need to collect examples from where to learn. This section presents the way in which an agent can proactively collect those examples and how can a competence model be learnt from them.

The proactive learning technique consists of several steps (shown in Figure 6): first, an agent A_i that wants to learn a competence model obtains a set of cases (that can be taken from its individual case base), and those cases are transformed to problems (by removing their solutions); the agent sends then those problems to other agents and obtains their individual predictions for those problems; with the predictions made by the other agents for all the problems sent, A_i will construct a set of voting situations; finally, these voting situations will be the input of a learning algorithm from which the competence models will be learnt.

Moreover, in order to easily apply standard machine learning techniques, we need to characterize the voting situations by defining a collection of attributes in order to express them as attribute-value vectors (since most machine learning techniques work with attribute-value vectors). The *characterization of a voting situation* $\mathcal{R}_{\mathcal{A}_t^c}$ is a tuple consisting of several attributes:

- The attributes $A_1, ..., A_n$ are boolean. $A_i = 1$ if $A_i \in \mathcal{A}_t^c$ (i.e. if A_i is a member of the current committee), and $A_i = 0$ otherwise.
- $S^c = Sol(\mathcal{S}, P, \mathcal{R}_{\mathcal{A}_t^c})$ is the candidate solution.
- $V^c = Ballot(S^c, \mathcal{A}_t^c)$ are the votes for the candidate solution.
- $V^r = (\sum_{S_k \in \mathcal{S}} Ballot(S_k, \mathcal{A}_t^c)) - V^c$ is the sum of votes for all the other solutions.
- $\rho = \frac{V^c}{V^c + V^r}$ is the ratio of votes supporting the candidate solution.

We will use $\upsilon = \langle A_1, ..., A_n, S^c, V^c, V^r, \rho \rangle$ to note the characterization of a voting situation. Moreover, an *M-example* m derived from a case c is a pair $m = \langle \upsilon, \omega \rangle$, where υ is the characterization of a voting situation $\mathcal{R}_{\mathcal{A}_t^c}$ and ω represents the "prediction correctness" of the voting situation, such that $\omega = 1$ if the candidate solution of the voting situation $\mathcal{R}_{\mathcal{A}_t^c}$ was the correct one (i.e. if $S^c = c.S$) and $\omega = 0$ otherwise (if $S^c \neq c.S$).

Therefore, a competence model $M \in \mathcal{M}_{A_i}$ will be learnt by collecting a set of *M-examples* to form a data set and learning the competence model from them using induction.

Figure 6 presents a scheme of the proactive learning process that will be explained in the remaining of this section. Specifically, the steps involved in the proactive learning process are the following ones:

1. An agent that wants to learn a competence model M, selects a set of cases from its individual case base.

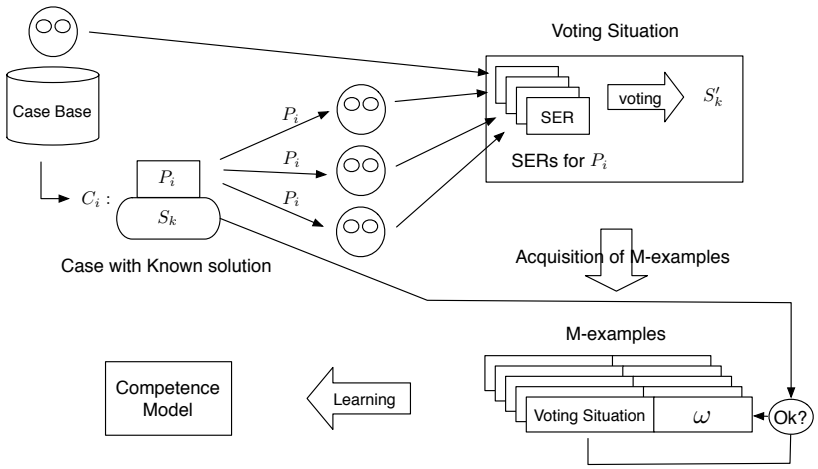


Figure 6. Detailed graphical representation of the proactive learning technique to learn competence models.

2. Those cases are transformed into problems by removing their solution and are sent to other agents in the $\mathcal{M}$AC system in order to obtain their individual predictions.

3. Voting situations are then built from these individual predictions, and from these voting situations, M-examples are constructed.

4. Finally, with the collection of M-examples, a competence model is learnt using an induction algorithm.

These four steps will be presented in detail in the rest of this section.

4.1. Acquisition of *M-examples*

In this section we are going to present the proactive process that an agent follows in order to acquire M-examples from where to learn the competence models.

Since an agent A_i needs to learn several competence models, a different training set T_M will be needed to learn each competence model $M \in \mathcal{M}_{A_i}$. We will call $\mathcal{T}_{A_i} = \{T_{M_c}, T_{M_{A_1}}, ...T_{M_{A_{i-1}}}, T_{M_{A_{i+1}}}, ..., T_{M_{A_n}}\}$ to the collection of training sets needed by an agent A_i to learn the competence models.

For example, when A_i is building M_c (the competence model of the committee), A_i sends a problem P to the rest of agents in the $\mathcal{M}$AC system. After receiving their predictions, A_i builds the voting situation resulting of putting together all the SERs built by the agents. Then, A_i uses the voting system to determine the candidate solution of that

voting situation. If the candidate solution for the problem P is correct, then A_i can build an M_c-example with $\omega = 1$, and if the prediction is incorrect, A_i can build an M_c-example with $\omega = 0$.

Specifically, an agent A_i that wants to obtain the collection of training sets needed to learn the competence models proceeds as follows:

1. A_i chooses a subset of cases $B_i \subseteq C_i$ from its individual case base.
2. For each case $c \in B_i$:
   a) A_i uses I_C (the interaction protocol of CCS) to convene a committee of agents $\mathcal{A}^c$ to solve the problem $c.P$. After this, A_i has obtained the SERs built by all the rest of agents in $\mathcal{A}^c$ for problem $c.P$.
   b) A_i solves $c.P$ using a *leave-one-out* method[1] and creates its own set of SERs $\mathcal{R}_{A_i}$.
   c) With the set $\mathcal{R}_{\mathcal{A}^c}$ of SERs obtained (that includes all the SERs from the other agents obtained in step (a) and the SERs of A_i computed in (b)), A_i builds a number of voting situations from where to construct M-examples (as explained below).

Notice that A_i can build more than one voting situation from the collection $\mathcal{R}_{\mathcal{A}^c}$ of SERs in Step 2.(c). For instance, the set of SERs built by A_i, $\mathcal{R}_{A_i} \subseteq \mathcal{R}_{\mathcal{A}^c}$ corresponds to a voting situation where only agent A_i has cast votes. The set of SERs built by A_i and any other agent A_j, $(\mathcal{R}_{A_i} \cup \mathcal{R}_{A_j}) \subseteq \mathcal{R}_{\mathcal{A}^c}$ corresponds to a voting situation where A_i and A_j have cast their votes. In the following, we will write $\mathcal{R}_{\mathcal{A}'}$ to refer to the set of SERs built by a set of agents $\mathcal{A}'$.

A *Valid Voting Situation* $\mathcal{R}_{\mathcal{A}'}$ for an agent A_i and a problem $c.P$ is a voting situation where A_i has casted its votes, i.e. a set of SERs built by a set of agents $\mathcal{A}'$ that at least contains A_i. Specifically, $\mathcal{R}_{\mathcal{A}'} \subseteq \mathcal{R}_{\mathcal{A}^c}$ such that $\mathcal{A}' \subseteq \mathcal{A}^c$ and $A_i \in \mathcal{A}'$.

Intuitively, a valid voting situation for an agent A_i is one in which A_i itself is a member of the committee. Therefore, a valid voting situation can be built by selecting the set of SERs built by any subset of agents $\mathcal{A}' \subseteq \mathcal{A}^c$ (such that $A_i \in \mathcal{A}'$). We can define the set of all the possible subsets of agents of $\mathcal{A}$ that contain at least A_i as $\mathbb{A}(A_i) = \{\mathcal{A}' \in \mathcal{P}(\mathcal{A}) | A_1 \in \mathcal{A}'\}$, where $\mathcal{P}(\mathcal{A})$ represents the parts of the set $\mathcal{A}$ (i.e. the

[1] The leave-one-out method works as follows: the agent A_i removes the case c from its case base; then it solves tries to solve $c.P$; after it has found a solution for $c.P$, the agent introduces c again into its case base. This method is usually used to test wether a learning system would be able to properly solve a given problem if the problem would not be present in its case base.

set of all the possible subsets of $\mathcal{A}$). Now it is easy to define the set of all the possible Valid Voting Situations for an agent A_i that can be constructed from $\mathcal{R}_{\mathcal{A}^c}$ as follows:

The *Set of Valid Voting Situations* for an agent A_i is: $\mathbb{V}(A_i) = \{\mathcal{R}_{\mathcal{A}'} | \mathcal{A}' \in \mathbb{A}(A_i)\}$, where $\mathcal{R}_{\mathcal{A}'}$ represents the set of SERs built by the set of agents $\mathcal{A}'$.

Using the previous definitions, we can decompose Step 2.(c) above in three sub-steps:

1. A_i takes a sample of all the possible Valid Voting Situations that can be built: $\mathbb{V}' \subseteq \mathbb{V}(A_i)$ (see below).
2. For every voting situation $\mathcal{R} \in \mathbb{V}'$, the agent A_i determines the characterization of the voting situation $\langle A_1, ..., A_n, S^c, V^c, V^r, \rho \rangle$.
3. With this characterization A_i can build M-examples. Specifically, A_i will build one M-example for each competence model $M \in \mathcal{M}_{A_i}$.

Let us now focus on how M-examples are constructed for each specific competence model $M \in \mathcal{M}_{A_i}$:

- To build an M_c-example, A_i determines the candidate solution $S^c = Sol(\mathcal{S}, c.P, \mathcal{R}_{\mathcal{A}'})$ obtained by applying the voting system to all the SERs in $\mathcal{R}_{\mathcal{A}'}$. If $Sol(\mathcal{S}, c.P, \mathcal{R}_{\mathcal{A}'}) = c.S$, then the following M_c-example is built: $m = \langle \langle A_1, ..., A_n, S^c, V^c, V^r, \rho \rangle, 1 \rangle$ where $\omega = 1$ because the M-example characterizes a voting situation where the predicted solution is correct. If $S^c \neq c.S$, then the following M_c-example is built: $m = \langle \langle A_1, ..., A_n, S^c, V^c, V^r, \rho \rangle, 0 \rangle$ where $\omega = 0$ because the M-example characterizes a voting situation where the predicted solution is not correct.
- To build an M_{A_j}-example, A_i determines the individual candidate solution yield by A_j, i.e. $S^c_{A_j} = Sol(\mathcal{S}, c.P, \mathcal{R}_{A_j})$. If $S^c_{A_j} = c.S$ (i.e. the prediction of A_j is correct), then the following M_{A_j}-example is built: $m = \langle \langle A_1, ..., A_n, S^c, V^c, V^r, \rho \rangle, 1 \rangle$ and if $S^c_{A_j} \neq c.S$ (i.e. the prediction of A_j is incorrect), then the following M_{A_j}-example is built: $m = \langle \langle A_1, ..., A_n, S^c, V^c, V^r, \rho \rangle, 0 \rangle$.

Notice that with each voting situation $\mathcal{R} \in \mathbb{V}'$, an M-example can be constructed for each different competence model in $\mathcal{M}_{A_i}$. Therefore, the larger the size of $\mathbb{V}' \subseteq \mathbb{V}(A_i)$, the larger the number of M-examples that can be constructed. The size of $\mathbb{V}(A_i)$ (that is equivalent to the size of $\mathbb{A}(A_i)$) depends on the number of agents in the committee convened to solve each of the problems $c.P$ (where $c \in B_i \subseteq C_i$). In fact, the size

of $\mathbb{V}(A_i)$ grows exponentially with the size of the set of convened agents: there are 2^{n-1} different Valid Voting Situations for a $\mathcal{M}$AC system with n agents. Therefore, building all the M-examples that can be derived from all possible valid voting situations in $\mathbb{V}(A_i)$ may be unfeasible or impractical. For that reason, an agent using the proactive learning technique to learn competence models will take a sample $\mathbb{V}' \subseteq \mathbb{V}(A_i)$ instead of considering all of them.

The number of M-examples that an agent builds for each competence model M is about $\#(B_i) \times \#(\mathbb{V}')$ (where the $\#(A)$ notation represents the number of elements of B, where A is any set). The number of M-examples that agents need to collect for learning appropriate competence models may vary in function of the application domain. In general, the more M-examples collected, the better, however collecting many M-examples will waste resources of the agent, thus in function of the resources an agent is wiling to spend in building competence models, the number of M-examples to collect must be determined. In our experiments we have imposed the limit of at most 2000 M-examples for each competence model. Therefore, the agents in our experiments will take subsets $\mathbb{V}' \subseteq \mathbb{V}(A_i)$ to have at most $2000/\#(B_i)$ voting situations. Moreover, in our experiments, an agent A_i using the proactive learning technique uses all the case base C_i as the set B_i (i.e. $B_i = C_i$) (in order to maximize the diversity in the set of voting situations built), and therefore the size of $\mathbb{V}'$ will be at most $2000/\#(C_i)$ (see [19] for the definition of a method to select subsets $\mathbb{V}' \subseteq \mathbb{V}(A_i)$ in a more informative way than doing a random selection).

4.2. Induction of the Competence Models

Once an agent A_i has collected enough *M-examples*, good competence models can be learnt. In our experiments we have used an induction algorithm based on decision trees [30].

A decision tree is a predictive model for a specific problem, i.e. given a new problem, a decision tree predicts its solution. A decision tree is composed of two kind of nodes: *decision nodes* and *leaf nodes*. A decision node contains an expression, and as many child nodes as possible values that expression might have. When using the decision tree to predict the solution of a given problem P, the expression of the decision node is evaluated for P, and the branch corresponding the the value obtained is followed. This procedure is repeated until we reach a leaf node. Leaf nodes contain the predicted solution. If while solving a problem P using a decision tree we reach a leaf labelled S_k, then the predicted solution for P will be S_k.

Moreover, since competence models predict confidence values (i.e. real numbers in the interval $[0, 1]$), standard decision trees cannot be directly used as confidence values (since they are thought for predicting class labels). For that reason, we are going to define a variation of decision trees that we call *confidence trees*, that will suit our needs. As decision trees, a confidence tree is a structure consisting on two types of nodes: *decision nodes* and *leaf nodes*. Decision nodes are identical to that of decision trees, however leaf nodes of confidence trees differ. A leaf node in a confidence tree contains three real numbers: p_l^-, p_l, and p_l^+ (such that $p_l^- \leq p_l \leq p_l^+$); where p_l is the expected confidence in that a voting situation that is classifier in a leaf l will yield a correct candidate solution, and p_l^- and p_l^+ are respectively, the pessimistic and optimistic estimations of that confidence.

In order to learn confidence trees, we will use a standard decision tree learning algorithm [30], but with the following considerations:

1. Numerical attributes are discretized. Each numeric attribute a is discretized to have just 2 possible values. The discretization is performed by computing a threshold κ. Left branch of the decision node will have the M-examples with $value(a) \leq \kappa$ and in the right branch all the M-examples which $value(a) > \kappa$.

2. Error-based pruning [5] of the tree is used to avoid overfitting (i.e. for not learning a too specific decision tree that overfits the particularities of the training set).

3. Usually, when learning a decision tree, the solution class of each leaf is decided in function of the solution of the examples of the training set that fall in that leaf. However, instead of deciding a single solution class for each leaf, we will store the amount of examples of each solution class that fall in each leaf. Figure 7.a shows a decision tree such that in each leaf l, the number of M-examples with $\omega = 1$ and with $\omega = 0$ is shown. For instance, you can see that in the right-most leaf, there are 457 examples with $\omega = 1$ and 29 examples with $\omega = 0$. That means that in the training set, there were 457 examples with $\rho > 0.70$ and $\omega = 1$, and 29 examples with $\rho > 0.70$ and $\omega = 0$.

Once a decision tree is learnt, we transform it into a confidence tree by maintaining decision nodes and transforming leaf nodes in the following way: Let a_l be the number of M-examples with $\omega = 1$ and b_l the number of M-examples with $\omega = 0$ in a given leaf l of the learnt decision tree. Then, the values of the corresponding leaf in the confidence tree take the following values:

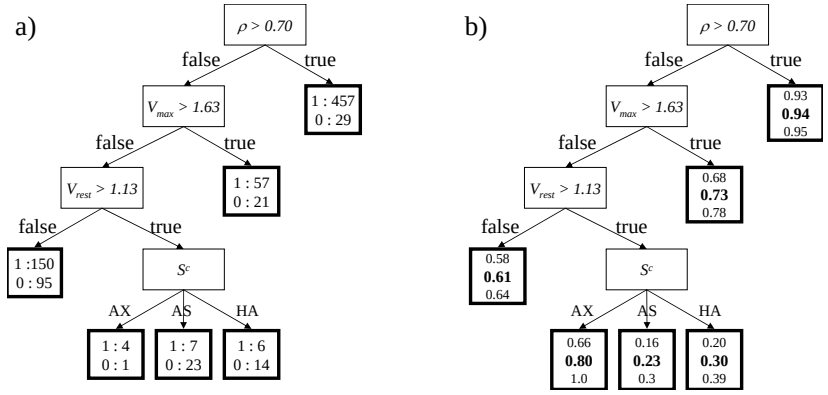


Figure 7. a) Decision tree learnt as the competence model M_c in a $\mathcal{M}$AC system composed of 5 agents. b) Confidence tree computed from the decision tree shown in a). For the numerical attributes, the right branches of each node contain the M-examples that match the condition in the node. AS, AS and HA are the possible solution classes in $\mathcal{S}$. The left figure shows the number of M-examples with each confidence value that have fallen in each tree, and the right figure shows the estimation of the confidence in each leaf.

- $p_l = (1/(a_l + b_l)) * (1 * a_l + 0 * b_l)$ is the expected confidence of an M-example classified in leaf l.
- p_l^-: the pessimistic estimation of the confidence of the confidence of an M-example classified in that leaf l (see below).
- p_l^+: the optimistic estimation of the confidence of the confidence of an M-example classified in that leaf l (see below).

Figure 7 shows an example of the conversion from a decision tree (on the left) to a confidence tree (on the right). On each leaf l of the confidence tree, the three values p_l^-, p_l, and p_l^+ (such that $p_l^- \leq p_l \leq p_l^+$) are shown. Since p_l is just an estimation of the confidence, if the number of M-examples in the leaf node l is small then p_l may be a poor estimator of the confidence of the candidate solution of voting situations classified on the leaf l. The greater the number of M-examples in leaf l, the better the estimation of the confidence. To solve this problem, instead of estimating the confidence as a single value, the agents will compute an interval, $[p_l^-, p_l^+]$, that ensures with 66% certainty that the real confidence value is in that interval. This interval depends on the number of examples in leaf l: the greater the number of M-examples, the narrower the interval will be (those intervals can easily computed numerically using basic bayesian probabilistic computations). In Figure 7.b, p_l^- and p_l^+ are shown above and below p_l respectively. For instance, if we look at the right most leaf in Figure 7 (the one with 457 M-

examples with confidence 1 and 29 M-examples with confidence 0), we can see that the estimated p_l is 0.94 and the interval is $[0.93, 0.95]$, a very narrow interval since the number of M-examples to estimate the confidence is high.

For the purposes that competence models will have in the dynamic committee collaboration strategies, pessimistic estimation is safer than any other estimation (expected p_l or optimistic p_l^+). Using pessimistic estimations the worst that can happen is that the committee convened to solve a problem is larger than in should be. However, if we make a more optimistic estimation of the confidence, (using the expected p_l or optimistic p_l^+ estimations) the convener agent may stop inviting agents too early, thus failing to correctly solve a problem more often. Therefore, since confidence trees will be used as competence models,p_l^- will be used as the output of the competence model, i.e. the output of a competence model M for a voting situation $\mathcal{R}_{\mathcal{A}^c}$ is $M(\mathcal{R}_{\mathcal{A}^c}) = p_l^-$, where l is the leaf of the confidence tree in which the voting situation $\mathcal{R}_{\mathcal{A}^c}$ has been classified.

The next section presents an exemplification of the proactive learning technique used to learn the confidence trees that will be used as the competence models in the Proactive Bounded Counsel Collaboration Strategy.

4.3. Exemplification

In order to clarify the M-example acquisition process, we will describe an exemplification with a system composed of 3 agents $\mathcal{A} = \{A_1, A_2, A_3\}$. Moreover, notice that the this section tries to exemplify only the M-example acquisition process, and not the use of PB-CCS.

The agent A_1 is collecting M-examples to learn the competence models needed in the Proactive Bounded Counsel Collaboration Strategy. A_1 should learn three competence models: $\mathcal{M}_{A_1} = \{M_c, M_{A_2}, M_{A_3}\}$.

For that purpose, A_1 has selected a subset $B_1 \subseteq C_1$ of cases from its individual case base C_1. All the cases in B_1 will be used to acquire M-examples. In the experiments presented in this paper the agents use the policy $B_i = C_i$ to select the subset B_i of cases, however in a real scenario where the case base C_i might be arbitrarily high this might have a high cost. For that reason we say that, in general, an agent selects a subset $B_i \subseteq C_i$ instead of using all the case base. However, if there are enough computational resources B_i should be as large as possible in order to acquire a good training set from where to learn the competence models.

For each case $c \in B_1$ the agent A_1 convenes a committee to solve the problem $c.P$. For instance, imagine that for a specific problem $c_1 =$

$\langle P_1, S_1\rangle$, both A_2 and A_3 accept to join the committee, and send the following SERs to A_1: A_2 sends $\mathbf{R}_2 = \langle S_1, 3, P_1, A_2\rangle$ and A_3 sends $\mathbf{R}_3 = \langle S_2, 1, P_1, A_3\rangle$. Finally, A_1 has built the SER $\mathbf{R}_1 = \langle S_1, 2, P_1, A_1\rangle$ using a leave-one-out method. Therefore, A_1 has collected the set of SERs $\mathcal{R}_{\mathcal{A}^c} = \{\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3\}$ from the set of agents $\mathcal{A}^c = \{A_1, A_2, A_3\}$.

There are 4 possible subsets of $\mathcal{A}^c$ that contain A_1, namely $\mathbb{A}(A_1) = \{\{A_1\}, \{A_1, A_2\}, \{A_1, A_3\}, \{A_1, A_2, A_3\}\}$. Assume that the agent A_1 chooses the collection $\mathbb{A}' = \{\{A_1\}, \{A_1, A_2\}, \{A_1, A_3\}\}$ of subsets of agents to build voting situations from where to construct M-examples (recall from Section 4.1 that in general all subsects cannot be selected, since for a large number of agents there are an exponential number of them)

From the first subset of agents $\mathcal{A}' = \{A_1\}$, the following voting situation $\mathcal{R}' = \{R_1\}$ is built. A_1 computes the attributes that characterize the voting situation $\mathcal{R}'$: $(1, 0, 0, S_1, 0.66, 0.00, 1.00)$. From this voting situation, the three following M-examples can be built:

- An M_c-example: $\langle(1, 0, 0, S_1, 0.66, 0.00, 1.00), 1\rangle$, since the candidate solution S_1 is the correct one.
- An M_{A_2}-example: $\langle(1, 0, 0, S_1, 0.66, 0.00, 1.00), 1\rangle$, since the SER of agent A_2 endorses the correct solution class S_1. It is important to understand that this M_{A_2}-example characterizes a situation where A_1 has voted, the candidate solution of the current committee (containing only A_1) is S_1 and A_2 has not yet joined the committee. A confidence value $\omega = 1$ means that in this situation A_2 has predicted the correct solution class S_1.
- An M_{A_3}-example: $\langle(1, 0, 0, S_1, 0.66, 0.00, 1.00), 0\rangle$, since the SER of agent A_3 endorses an incorrect solution class S_2. As in the previous situation, this M_{A_3}-example characterizes a situation where A_1 has voted, the candidate solution of the current committee (containing only A_1) is S_1 and A_3 has not yet joined the committee. A confidence value $\omega = 0$ means that in this situation A_3 has predicted an incorrect solution class.

From the second subset of agents $\mathcal{A}' = \{A_1, A_2\}$, the following voting situation $\mathcal{R}' = \{R_1, R_2\}$ is built. The characterization is $(1, 1, 0, S_1, 1.41, 0.00, 1.00)$, and the M-examples that can be built are:

- An M_c-example: $\langle(1, 1, 0, S_1, 1.41, 0.00, 1.00), 1\rangle$, since the candidate solution S_1 is the correct one.
- An M_{A_3}-example: $\langle(1, 1, 0, S_1, 1.41, 0.00, 1.00), 0\rangle$, since the SER of agent A_3 endorses an incorrect solution class S_2.

Notice that no M_{A_2}-example is built from this voting situation, since A_2 is already a member of the committee corresponding to the characterized voting situation.

Finally, from the third subset of agents $\mathcal{A}' = \{A_1, A_3\}$, the following voting situation $\mathcal{R}' = \{R_1, R_3\}$ is built. The characterization is $(1, 0, 1, S_1, 0.66, 0.50, 0.57)$, and the M-examples that can be built are:

- An M_c-example: $\langle(1, 0, 1, S_1, 0.66, 0.50, 0.57), 1\rangle$, since the candidate solution S_1 is the correct one.
- An M_{A_2}-example: $\langle(1, 0, 1, S_1, 0.66, 0.50, 0.57), 1\rangle$, since the SER of agent A_2 endorses the correct solution class S_1.

Therefore, with just a single case $c \in B_i$, the agent A_i has built 3 M_c-examples, 2 M_{A_2}-examples and 2 M_{A_3}-examples. After A_1 has collected M-examples using all the cases in B_i, 3 training sets will be built: T_{M_c}, $T_{M_{A_2}}$, and $T_{M_{A_3}}$. From these 3 training sets, A_1 can now induce the corresponding confidence trees to be used as the competence models M_c, M_{A_2}, and M_{A_3}. Similarly, agents A_2 and A_3 can also use the same technique to acquire their respective competence models if they need them. Notice that each agent in a $\mathcal{M}$AC system is free to use the collaboration strategies and decision policies that it prefers. Therefore, if A_1 uses the proactive learning technique to learn its own competence models, A_2 and A_3 are not forced to use it. Each agent could acquire its competence models using another strategy or using PB-CCS with different parameter settings.

4.4. Proactive Bounded Counsel Cost

Autonomous learning of competence models has a computational cost for the agents. Specifically, the agents will have to send a set of problems to the other agents (in order to evaluate them), and also solve individually another set of problems (in order to build the committee competence model). However, notice that if learning is not used, the cost of learning the competence models does not disappear, but it is merely shifted to a previous phase where a good competence model (or any other policy to decide how to form committees) is built by hand.

Moreover, in a real system, the cost of acquiring *M-examples* to learn the competence models can me greatly lowered. Imagine that a specific agent wants to learn a competence model; the agent can solve the problems that arrive from users convening committees using any other collaboration strategy (such as CCS), and store the predictions that the other agents make during the solution of problems that arrive. Once the agent has collected enough experience, the competence model

can be learnt. In other words, it is not necessary for an agent to collect all the required *M-example* in one step, but they can be automatically acquired by interacting with other agents by solving real problems send by users.

5. Bounded Counsel Collaboration Strategy

In this section we are going to define a non-learning approach to form dynamic committees, the *Bounded Counsel Collaboration Strategy* (B-CCS). B-CCS works basically in the same way than PB-CCS, but uses predefined competence models instead of learnt ones. Thus B-CCS is only presented for comparison purposes, with the goal of evaluating the learnt competence models used by PB-CCS. The Bounded Counsel collaboration strategy is composed by an interaction protocol and two decision policies:

DEFINITION 5.1. *The* Bounded Counsel Committee Collaboration Strategy *(B-CCS) is a collaboration strategy* $\langle I_B, D_H, D_V \rangle$*, where* I_B *is the B-CCS interaction protocol shown in Figure 4,* D_H *is the Bounded Counsel* Halting *decision policy (used to decide when to stop inviting agents to join the committee), and* D_V *is the voting decision policy based on BWAV (see Section 2.4).*

B-CCS uses I_B, the same protocol as PB-CCS. Moreover, when a new agent is invited to join the committee in B-CCS, a random agent A_j is selected from the set of agents that do not belong to the committee. Thus, B-CCS requires only an individual decision policy: the Bounded Counsel *Halting* decision policy D_H, that decides whether inviting more agents to join the committee is needed.

The D_H decision policy uses the *C-Competence* model that measures the confidence in a solution predicted by a committee to be correct.

$$\textit{C-Competence}(\mathcal{R}^c) = \begin{cases} \frac{1}{M} Ballot(Sol(\mathcal{S}, \mathcal{A}^c), \mathcal{A}^c) & \text{If } N > 1, \\ min(Ballot(Sol(\mathcal{S}, \mathcal{A}^c), \mathcal{A}^c), 1) & \text{If } N = 1. \end{cases}$$

where $\mathrm{M} = \sum_{S_k \in \mathcal{S}} Ballot(S_k, \mathcal{A}^c)$, is the sum of all the votes casted by the agents and $\mathrm{N} = \#(\{S_k \in \mathcal{S} | Ballot(S_k, \mathcal{A}^c) \neq 0\})$, is the number of different classes for which the agents have voted for.

That is to say, if the agents in $\mathcal{A}^c$ have built SERs for a single solution ($N = 1$), the *Committee-Competence* model will return the ballot for that solution. Moreover, notice that the ballot for a solution when there are more than one agent in $\mathcal{A}^c$ can be greater than 1. Therefore we take

the minimum between the ballot and 1 to ensure that the competence models output confidence values within the interval $[0, 1]$. The intuition is that the higher the ballot, the larger the number of cases retrieved by the agents endorsing the predicted solution, and therefore the higher the confidence on having predicted the correct solution. Moreover, if the agents in $\mathcal{A}^c$ have built SERs for more than one solution (and therefore $N > 1$), the *C-Competence* model will return the fraction of votes that that are given to the most voted solution $Sol(\mathcal{S}, \{A_i\})$. The larger fraction of votes for the predicted solution, the larger the number of agents that have voted for the predicted solution or the larger the number of cases that each individual agent has retrieved endorsing the predicted solution, and therefore the higher the confidence on having predicted the correct solution.

Using this competence model, we can now define the D_H as a boolean decision policy that decides whether the convener agent can stop inviting agents to the committee; if $D_H(\mathcal{R}^c) = true$, no more agents will be invited to the committee.

$$D_H(\mathcal{R}^c) = (\textit{C-Competence}(\mathcal{R}^c) \geq \eta)$$

where η is a threshold parameter.

The intuition behind the D_H decision policy is that if the confidence on the solution predicted by the current committee is high enough, there is no need for inviting more agents to join the committee. Notice that when A_i is alone (and can be considered as a committee of 1) this decision is equivalent to choose between solving the problem individually or convening a committee. In our experiments we have set $\eta = 0.75$.

6. Experimental Evaluation

This section presents the experimental evaluation of the performance of PB-CCS. To evaluate the behavior of PB-CCS using the learnt competence models we have compared it against the Committee Collaboration Strategy (CCS) and the Bounded Counsel Collaboration Strategy (B-CCS). We have made experiments with $\mathcal{M}\mathsf{AC}$ systems composed of 3, 5, 7, 9, 11, 13, and 15 agents. Moreover, the agents use a standard 3-Nearest Neighbor (3-NN) [7] method to solve problems. Notice that we could have used a more complex CBR method, but we have chosen the standard 3-NN since our goal is to analyze the behavior of PB-CCS and not to obtain the maximum classification accuracy. We have designed an experimental suite with a case base of 280 marine sponges pertaining to three different orders of the *Demospongiae*

class (*Astrophorida*, *Hadromerida* and *Axinellida*). In an experimental run, training cases are randomly distributed among the agents. In the testing stage unknown problems arrive randomly to one of the agents. The goal of the agent receiving a problem is to identify the correct biological order given the description of a new sponge. Moreover, all the results presented here are the result of the average of five 10-fold cross validation runs.

Moreover, in order to investigate whether the the proactive learning technique used in PB-CCS learns adequate competence models under different circumstances, we have performed experiments in three different scenarios: the *uniform* scenario, the *redundancy* scenario, and the *untruthful agents* scenario.

Uniform: in this scenario each individual agent receives a random sample of the training set without replication of cases (i.e. the case bases of the agents are disjoint).

Redundancy: in this scenario each agent receives a random sample with replication of cases (i.e. two agents may own the same case). To measure the degree of redundancy introduced, we will define the redundancy index R as follows:

$$R = \frac{(\sum_{i=1...n} \#(C_i)) - N}{N * (n-1)}$$

where n is the number of agents, $N = \#(\cup_{i=1...n} C_i)$ is the total number of different cases in the system, and C_i is the individual case base of the agent A_i.

When the individual case bases of the agents are disjoint, there is no redundancy at all, thus $R = 0$. Notice that this is true since when the case bases are disjoint $N = \#(\cup_{i=1...n} C_i) = \sum_{i=1...n} \#(C_i)$; thus the numerator is zero. Moreover, when all the individual case bases of the agents are identical (all the agents own the same cases) the redundancy is maximal, and thus $R = 1$. Notice that this is also true since if all the case bases are identical, then $\forall_j \cup_{i=1...n} C_i = C_j$, and thus $\#(C_j) = N$. For that reason, $\sum_{i=1...n} \#(C_i) = N \times n$; therefore $R = 1$.

In our experiments we have used a degree of redundancy of $R = 0.1$, and the data set that is distributed among the agents has 280 cases (as we perform a 10 fold cross validation, there training set to distribute among the agents at each fold has 254 cases). To have an idea of what $R = 0.1$ represents, consider this: in a 5 agents scenario with $R = 0.0$ each agent will receive about 50.4 cases (since the 280 cases in the data set are divided in a training set of 252 cases and a test set of 28 cases during the 10 fold cross validation, and 252 / 5 = 50.4). Moreover, with $R = 0.1$, each agent will receive 70.54 cases since some of the training cases will be replicated among the agents case bases (if $R = 1.0$ each

agent will receive the 252 training cases). In a 9 agents scenario, with $R = 0.0$ each agent will receive about 28.00 cases, and with $R = 0.1$ each agent will receive 50.4 cases in average.

Untruthful Agents: in this scenario some of the agents in the committee are untruthful, i.e. when an agent asks them for help, they will sometimes answer a solution different from their real individual prediction (i.e. they lie). However, those agents answer the truthful solution when they are in the role of the convener agent.

The goal of performing experiments in these scenarios is to test whether the individually learnt competence models are useful to decide when to stop inviting agents to join the committee and which agents to invite under different conditions. The uniform scenario is the basic scenario, where each individual agent has a different sample of the training set. Moreover, since each agent has more cases in the redundancy scenario than in the uniform scenario, it is expected that each individual agent has a greater individual accuracy. Therefore, we expect that the number of times an agent solves a problem individually without need to convene a committee increases in the redundancy scenario. Moreover, the average number of agents needed to solve a problem should decrease for the same reason.

Finally, the untruthful agents scenario models a situation in which not all the agents of the system can be trusted. We have designed this scenario to test whether the learnt competence models can detect which agents in the system can be trusted and which cannot. In this scenario, we expect that the performance of the committee decreases with respect to the uniform scenario. Moreover, by using competence models, the proactive bounded counsel collaboration strategy should be able to detect untruthful agents and very seldom invite them to join the committee; consequently we expect the performance of the proactive bounded counsel collaboration strategy (PB-CCS) not to decrease as much as the performance of the fixed committee (CCS), thus showing a more robust behavior.

These three scenarios are evaluated on a single data set. Using several data sets would not add any more meaningful information; the only apparent difference between several data sets is the degree in which the ensemble effect increases the committee accuracy. However, this is not a primary concern here, since our goal is evaluating the performance of the dynamic committees with respect to convening always the full committee in a given data set.

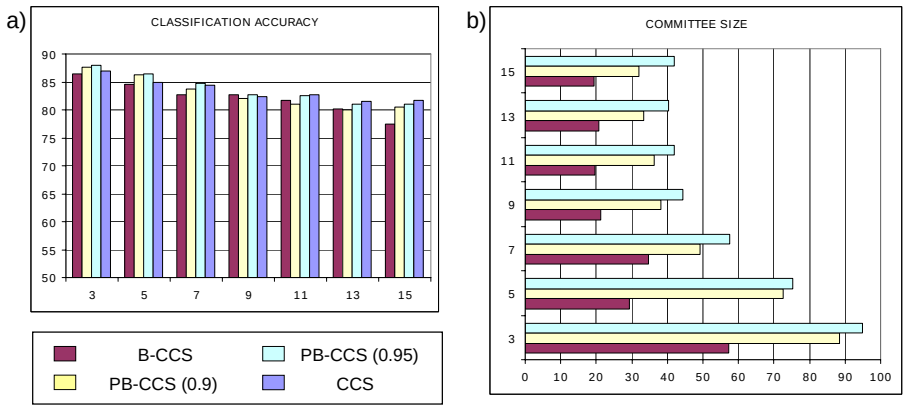


Figure 8. Classification accuracy and average committee size for agents using CCS, B-CCS, and PB-CCS in the sponges data set and using 3-NN in the uniform scenario.

6.1. PB-CCS Evaluation in the Uniform Scenario

Figure 8 shows the results for the uniform scenario. Specifically, Figure 8.a shows the classification accuracy and Figure 8.b shows the average committee size. $\mathcal{M}$AC systems with 3, 5, 7, 9, 11, 13 and 15 agents are tested. For each $\mathcal{M}$AC system results for agents using CCS, B-CCS, and PB-CCS are presented. Moreover, two different parameter settings have been evaluated for PB-CCS: the first one with $\eta_1 = 0.9$ and $\eta_2 = 0.5$ and the second one with $\eta_1 = 0.95$ and $\eta_2 = 0.5$. In the first parameter settings the convener agent will request a confidence of at least 0.9 in order to stop inviting agents to join the committee, and in the second parameter settings, the convener agent will request a confidence of at least 0.95. Therefore, the expected behavior is that in the second parameter settings both the convened committees and the classification accuracy would be larger. Moreover, both parameter settings request that all invited agents have at least a confidence of 0.5 of predicting the correct solution for the current problem.

Before analyzing the results shown in Figure 8, notice that as the number of agents increase, each agent receives a smaller case base. Thus, the classification accuracy of each individual agent is lower in the experiments with many agents. The effect of this is that the accuracy of all the collaboration strategies diminishes as the number of agents increases. However, it is important to note that this is not due to the number of agents, but to the way in which experiments have been performed, since in our experiments a larger number of agents implies smaller case bases (since the training set is divided among all the agents in the system and therefore, the more agents, the less cases that each agent receives).

Figure 8 shows that the classification accuracy of PB-CCS is very close to that of CCS. In fact, with $\eta_1 = 0.95$ the difference in classification accuracy between PB-CCS and CCS is not statistically significant. Moreover, the classification accuracy of PB-CCS (both with $\eta_1 = 0.9$ and $\eta_1 = 0.95$) is higher than the classification accuracy of B-CCS in all of the $\mathcal{MAC}$ systems except in the 9 agents system (where the difference is not statistically significant).

Figure 8.b shows the average size of the committees convened by PB-CCS and B-CCS expressed as the percentage of the agents in the $\mathcal{MAC}$ system convened in average (we do not show the size of the committees convened by CCS that is always 100% since CCS invites all the agents to join the committee). The figure shows that the average size of the committees convened by PB-CCS is smaller than the committees convened by CCS and specially in $\mathcal{MAC}$ systems with a large number of agents. The figure also shows that the average size of the committees convened by PB-CCS is larger than in B-CCS. In fact, PB-CCS invites more agents to join the committee when needed (since PB-CCS has a higher classification accuracy than B-CCS). Moreover, the threshold parameter η_1 affects the average size of the committee: if $\eta_1 = 0.95$ the size of the committees tends to be larger than with $\eta_1 = 0.9$, as expected.

Therefore PB-CCS achieves a better tradeoff of accuracy and committee size than CCS since the classification accuracy achieved by PB-CCS with $\eta_1 = 0.95$ is undistinguishable of the accuracy of CCS while the average size of a committee convened by PB-CCS is much smaller than 100% (the size of a committee convened by CCS). B-CCS also achieves an interesting tradeoff of accuracy and committee size: it achieves classification accuracy that are only a bit lower than that of the committee, and the committee size is much smaller than that of the committee. Notice that the only difference between PB-CCS and B-CCS is that in PB-CCS agents learn their own competence models, and in B-CCS competence models have to be predefined. The competence models used by B-CCS in these experiments have been hand-tuned for the uniform scenario, and thus B-CCS performs really well. However, as we will see in the next sections, the problem of B-CCS is that its competence models have to be defined (by a human user) for each different scenario in which an agent has to operate, while agents using PB-CCS can be left alone in any scenario and they will learn their own competence models without needing the intervention of a human user.

Figure 9 shows the percentage of times that the convener agent has convened committees of different sizes with $\eta_1 = 0.9$. An horizontal bar is shown for each $\mathcal{MAC}$ system. Each bar is divided in several intervals: the leftmost interval represents the percentage of times that the

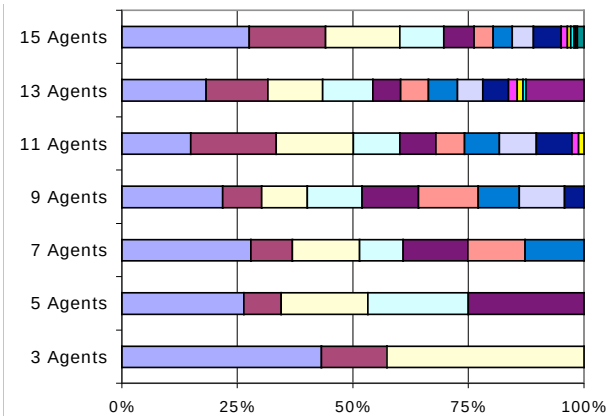


Figure 9. Percentage of times that the convener agent has convened committees of different sizes in the uniform scenario using PB-CCS with $\eta_1 = 0.9$.

convener agent has solved the problem individually; the second interval represents the percentage of times that a committee of 2 agents has been convened, and so on. The right most interval represents the percentage of times that a committee containing all the agents in the system has been convened. Figure 9 shows that in the 3 agents system, about 40% of the times the convener agent solves the problem individually without the need of convening a committee. However, this percentage is reduced in the $\mathcal{M}$AC systems with more agents; this is an expected result since in systems with more agents individual case bases are smaller and the individual accuracy is lower; consequently, the Proactive Bounded Counsel Halting decision policy D_H decides more often to convene a committee. However, even for a 15 agents system, more than 25% percent of the times an agent can solve problems individually without compromising the overall $\mathcal{M}$AC performance. This shows that even with a large number of agents (where each agent has a small case base) the decision policies are able to detect that there are problems that can be solved individually without reducing the classification accuracy. This is another evidence that the proactive learning is able to learn adequate competence models.

Summarizing, PB-CCS in the uniform scenario can achieve a classification accuracy undistinguishable to that of CCS but convening smaller committees. Consequently we can conclude that the proactive learning process is producing adequate competence models (since they exhibit the expected behavior). Moreover, we have seen that varying parameters η_1 and η_2 have the expected result in the behavior of PB-CCS since $\eta_1 = 0.95$ achieves a higher accuracy than $\eta_1 = 0.9$. The next section analyzes the behavior of PB-CCS in a different scenario.

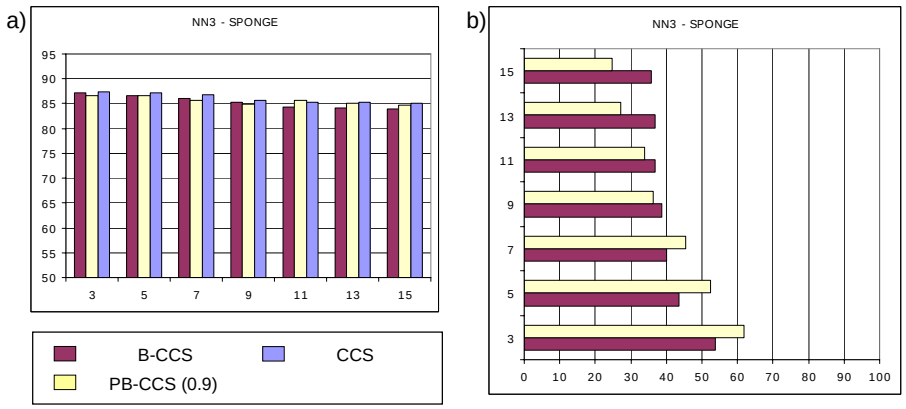


Figure 10. Classification accuracy and average committee size for agents using CCS, B-CCS, and PB-CCS in the sponges data set and using 3-NN in the redundancy scenario.

6.2. PB-CCS Evaluation in the Redundancy Scenario

In the redundancy scenario the case bases of the individual agents are not disjoint as in the uniform scenario, but have some overlapping, i.e. there are cases that are present in more than one agents' case base. This may interfere in the proactive learning process, since if two agents have a large intersection between their case bases the competence models that they learn about each other could be overestimating their real confidence. Moreover, we have used $\eta_1 = 0.9$ and $\eta_2 = 0.5$ for all the experiments in the redundancy scenario.

Figure 10 shows the results for the redundancy scenario. Figure 10.a shows that the classification accuracy of PB-CCS, B-CCS, and CCS are very similar, and their accuracy values are higher than those achieved in the uniform scenario. In fact, the difference in classification accuracy is only statistically significant in the 11, 13, and 15 agents systems where B-CCS achieves a lower classification accuracy than PB-CCS and CCS. Therefore, PB-CCS is as proficient as CCS.

In terms of committee size, PB-CCS convenes much smaller committees than the 100% committee of CCS as Figure 10.b shows. Again, this is specially noticeable in $\mathcal{MAC}$ systems with a large number of agents. For instance, in a $\mathcal{MAC}$ system with 13 agents, less than the 30% of the agents are convened in average, while CCS always convenes the 100% of the agents. Comparing the behavior of the dynamic committee strategies in the redundancy scenario with their behavior in the uniform scenario, it would be expected that they convene smaller committees in the redundancy scenario since individual agents have higher classification accuracy. PB-CCS shows exactly this behavior, i.e.

it convenes smaller committees in the redundancy scenario. However, B-CCS convenes larger committees in the redundancy scenario than in the uniform scenario. This happens because the competence models used by B-CCS are fixed, and do not change from one scenario to the other. This shows that learning competence models, as PB-CCS does, instead of using predefined ones, as B-CCS does, is a clear advantage. Moreover, another effect that we expect is that the classification accuracy of the collaboration strategies is higher in the redundancy scenario since the accuracy of the individual agents is higher. Comparing Figure 8 with Figure 10 we can observe that the three collaboration strategies show this behavior and their accuracy in the redundancy scenario is higher than in the uniform scenario.

Concerning the behavior of B-CCS, Figure 10 shows an interesting fact. As we previously said, B-CCS uses predefined competence models that in these experiments where hand-tuned to perform well in the uniform scenario. Figure 10 clearly shows that the behavior of B-CCS degrades as the number of agents increase (i.e. B-CCS achieves lower classification accuracy compared with PB-CCS or CCS and convenes larger committees than PB-CCS as the number of agents increase). This effect has a clear explanation. In the experiments in the redundancy scenario, we have fixed a redundancy index of $R = 0.1$; however $R = 0.1$ does not represent the same amount of redundancy in the 3 agents system than in the 15 agents system. In fact, a redundancy index of $R = 0.1$ in a 15 agents system is a huge degree of redundancy. Therefore, the larger the number of agents in the redundancy scenario, the further we are from the uniform scenario, and thus the further we are from the scenario for which the competence model of B-CCS was designed (and thus, the worse B-CCS performs).

Finally, Figure 11 shows the percentage of times that the convener agent has convened committees of different sizes in the redundancy scenario. Figure 11 shows that in the redundancy scenario, agents using PB-CCS solve problems individually more often than in the uniform scenario (shown in Figure 9). Therefore the proactive learning process has acquired good competence models, since the behavior of PB-CCS is the expected one, i.e. convenes smaller committees in the redundancy scenario since since if the individual accuracy is higher, the agents will individually solve problems correctly more often, and therefore, a committee has to be convened less often (and if there is the need to convene one, it can be convened with a smaller number of agents). For instance, in $\mathcal{M}$AC systems composed of 9 agents or less, agents solve problems individually between a 40% and a 50% of the times and in systems with 11 agents or more, in the 50% of the times no more than 2

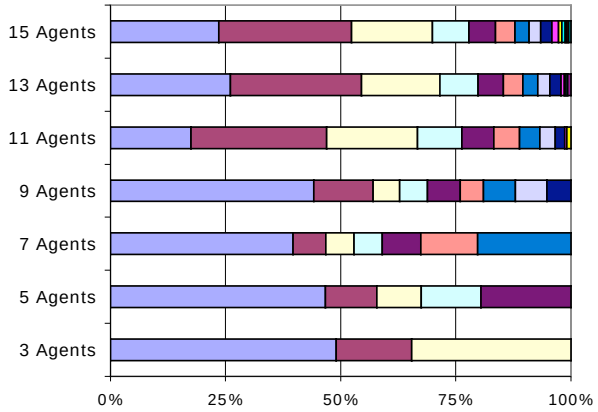


Figure 11. Percentage of times that the convener agent has convened committees of different sizes in the redundancy scenario using PB-CCS with $\eta_1 = 0.9$.

agents are invited to join the committee, while in the uniform scenario more agents were needed in average.

Summarizing, we have seen that redundancy improves individual accuracy and PB-CCS is able to detect that since it convenes smaller committees. Moreover, we have also seen that redundancy improves the accuracy of CCS and also that of PB-CCS(even convening smaller committees).

6.3. PB-CCS Evaluation in the Untruthful Agents Scenario

The untruthful agents scenario has two goals: the first one is to evaluate the robustness of PB-CCS in the presence of malicious agents (that is equivalent to evaluate the robustness of PB-CCS to noise); the second goal is to evaluate whether the proactive learning process produces adequate competence models, i.e. competence models that can detect that there are some agents that have a very low confidence (the untruthful agents).

Specifically, we have prepared a scenario where some agents in the $\mathcal{MAC}$ system will lie in their predictions when forming part of a committee. These *untruthful* agents will tell their individually predicted solution truthfully when they are convener agents, but will sometimes lie to other conveners. In our experiments we have set to 50% the probability of an untruthful agent to lie about its individual prediction. Specifically, there will be 1, 2, 3, 4, 5, 6 and 7 untruthful agents in the 3, 5, 7, 9, 11, 13 and 15 agents systems respectively. Moreover, in this scenario we expect that the Proactive Bounded Counsel Agent Selection decision policy, D_{AS}, is able to effectively decide which agents have a

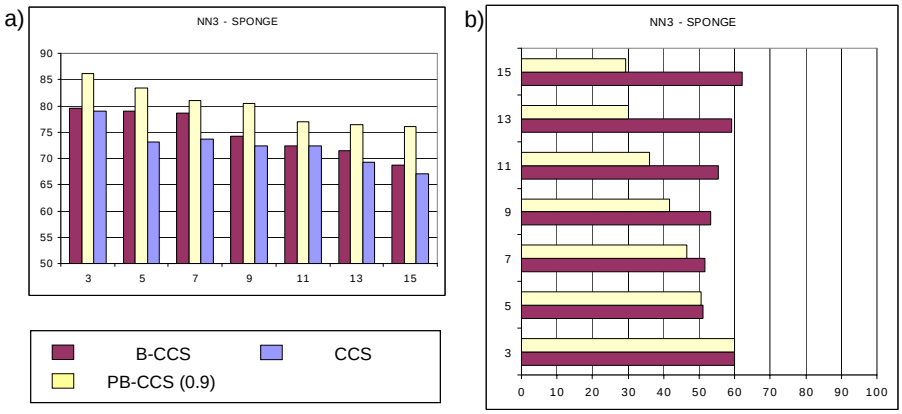


Figure 12. Classification accuracy and average committee size for agents using CCS, B-CCS, and PB-CCS in the sponges data set and using 3-NN in the untruthful agents scenario.

high confidence and which ones have a low confidence, so that untruthful agents are very seldom invited to join a committee. Finally, in the presence of the untruthful agents, it is expected that the classification accuracy of all the collaboration strategies is lower than in the uniform or redundancy scenarios since there are less agents with high confidence in the system that can be invited to join the committee.

Figure 12 shows the results for the untruthful agents scenario. The threshold parameters are set to $\eta_1 = 0.9$ and $\eta_2 = 0.5$. Figure 12.a shows that in this scenario the accuracy achieved by CCS and B-CCS is lower than the accuracy achieved by PB-CCS (in fact, the accuracy of CCS is even lower than the accuracy achieved by B-CCS since CCS always invites the untruthful agents to join the committee, while B-CCS does not). Moreover, comparing the accuracy achieved by the three collaboration strategies in the untruthful agents scenario with that achieved in the uniform scenario (shown in Figure 8) we see that they all achieve lower accuracy in the untruthful agents scenario. This decrease of classification accuracy is expected, since the presence of untruthful agents leaves less truthful agents to form committees with, and thus the maximum accuracy that can be reached is lower.

Since CCS does not perform any agent selection, all the untruthful agents are convened and its accuracy drops from 81.71% to 66.80% in the 15 agents scenario. Thus, we can conclude that CCS is not robust when there are agents that cannot be trusted. B-CCS selects agents randomly, and thus also convenes untruthful agents too often, resulting in a decreased classification accuracy. However, PB-CCS does use an agent selection policy, and as Figure 12.a shows, the accuracy of PB-CCS is much higher than that of B-CCS and CCS. This shows

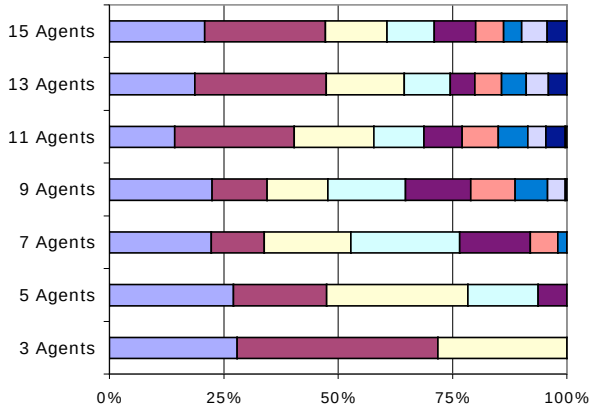


Figure 13. Percentage of times that the convener agent has convened committees of different sizes in the untruthful scenario using PB-CCS with $\eta_1 = 0.9$.

that PB-CCS is much more robust in the presence of untruthful agents than CCS and B-CCS. Moreover, the accuracy of PB-CCS drops (with respect to the uniform scenario) because there are less agents to convene committees with, and not because of a bad agent selection policy, as we will later show.

Concerning the committee sizes, Figure 12.b shows that the average size of the committees convened by PB-CCS is smaller than those convened by B-CCS. As the number of agents increase, the difference in size of the committees convened by B-CCS and PB-CCS increases. The explanation is that PB-CCS uses learnt competence models in the D_{AS} decision policy to select which of the other agents is the best one to be invited to join the committee, and thus untruthful agents are very seldom invited to join committees. Results concerning accuracy and average committee size in Figure 12 prove that this decision policy is useful and that effectively helps to convene a better committee than those convened using B-CCS or CCS. Consequently, this proves that the proactive learning process produces adequate competence models since the decision policy that uses them behaves as we would expect. In contrast, B-CCS uses a random decision policy to determine which agents are invited to join the committee, and therefore, untruthful agents are regularly invited to the committee. An untruthful agent that joins a committee will not likely contribute to increase the confidence of the predicted solution, and more agents will need to be invited, thus increasing the average committee size.

Figure 13 shows the percentage of times that the convener agent has convened committees of different sizes in the untruthful agents scenario. Specifically, we see that agents using PB-CCS in the untruth-

Table I. Average number of times that truthful and untruthful agents are invited to join a committee.

Agents	3	5	7	9	11	13	15
Truthful	47.57%	44.14%	43.63%	31.0%	32.0%	32.75%	31.8%
Untruthful	5.07%	9.03%	6.82%	7.14%	9.14%	11.57%	11.08%

ful agents scenario tend to convene smaller committees than in the uniform scenario (Figure 9). Committees convened by PB-CCS in the untruthful agents scenario are smaller because there are less agents with a high confidence that can be invited to join the committee. In fact, agents in the untruthful agents scenario should solve problems individually (without convening a committee) with the same frequency than agents in the uniform scenario, but the learnt competence models will detect that there is a subset of agents with low confidence and they will very seldom be invited to join the committee. For instance, in the 15 agents scenario, PB-CCS never convenes committees with more than 10 agents. Moreover, Figure 13 shows that agents in the untruthful agents scenario solve problems individually more or less the same percentage of times as in the uniform scenario (except for the 3 agents system).

For the purpose of assessing the degree in which the Proactive Bounded Agent Selection decision policy D_{AS} is able to detect the untruthful agents, the number of times that each agent has been invited to join a committee has been counted, summarized in Table I. For each $\mathcal{MAC}$ system, two values are shown: the average number of times that a truthful agent has been convened to a committee and the average number of times that an untruthful agent has been convened to a committee. For instance, in the 3 agents $\mathcal{MAC}$ system, each one of the two truthful agents is invited to join a committee a 47.57% of the times while the only untruthful agent is only invited to join a committee 5.07% of the times. This clearly shows that D_{AS} selects a truthful agent much more often. In fact, the degree to which D_{AS} is able to detect the untruthful agents depends of the threshold parameter η_2. In these experiments we have set $\eta_2 = 0.5$, but if we set a higher value (e.g. $\eta_2 = 0.75$) untruthful agents would be invited even less often. Notice that $\eta_2 \geq 0.5$ in order to preserve one the preconditions of the ensemble effect, namely that the individual error of the individual classifiers must be lower than 0.5.

The conclusion that we can draw form the experiments in the untruthful agents scenario is that PB-CCS is more robust than CCS and that B-CCS when the assumption that all the agents in the system are truthful does not hold, i.e. when not all the agents can be trusted. The result is that PB-CCS achieves a higher classification accuracy than both CCS and B-CCS and also convenes smaller committees.

7. Related Work

Three main areas are related to our work: ensemble learning, distributed CBR, and team formation.

Concerning ensemble learning, the "ensemble effect" is a general result on multiple model learning [15], that demonstrated that if uncorrelated classifiers with error rate lower than 0.5 are combined then the resulting error rate must be lower than the one made by the individual classifiers. The BEM (*Basic Ensemble Method*) is presented in [27] as a basic way to combine continuous estimators, and since then many other methods have been proposed: *Stacking generalization* [31], *Cascade generalization* [13], *Bagging* [4] or *Boosting* [12] are some examples. However, ensemble methods assume a centralized control of all the data while this is not true in our approach. Ensemble methods assume that all data is available to a centralized algorithm that constructs the individual classifiers that form the ensemble. In our approach each agent is the owner of its individual data (each individual agent has only access to the data contained in its own case base, and has no access to the data in other agents' case bases), and the distribution of data among agents cannot be determined using a centralized algorithm. Moreover, the control in $\mathcal{M}$AC systems is decentralized, and the global effect is achieved by individual decisions taken by the agents, while in ensemble learning all the decisions are made in a centralized way.

The meta-learning approach in [6] is applied to partitioned data set. They experiment with a collection of classifiers which have only a subset of the whole case base and they learn new meta-classifiers whose training data are based on predictions of the collection of (base) classifiers. They compare their meta-learning approach results with weighted voting techniques. The final result is an *arbitrator tree*, a centralized method whose goal is to improve classification accuracy. This approach is slightly more similar to ours than other techniques in ensemble learning, since the base assumption is that there exist a set of base classifiers (that are not created by the ensemble method), and the meta-learning approach just learns a way to combine their predictions (although in a centralized way).

Another related area is that of distributed CBR systems. McGinty and Smyth [18] present collaborative case-based reasoning (CCBR) as a framework where experience is distributed among multiple CBR agents. Their individual agents are only capable of solving problems that fall within their area of expertise. When an agent cannot solve a problem, it broadcasts the problem to the rest of agents, and if there is some agent capable of solving it, it will return the relevant retrieved cases to the initial agent. This approach differs from ours in the sense that they only perform case retrieval in a distributed way. The initiating agent receives all the relevant cases contained in all the case bases of the other agents, and then it solves the problem locally. In our approach, an agent can only work with its individual case base since no agent has access to the case base of another agent. Thus, while the CCBR approach can be seen as a distributed-retrieval approach, our approach can be seen as a distributed-reuse approach. Another related approach is *multi-case-base reasoning* (MCBR) [16, 17]. MCBR deals with distributed systems where there are several case bases available for the same task. Moreover, each case base may not correspond to exactly the same problem, or may reflect some different user preferences, etc. Therefore cases must be adapted to be moved from one case base to another. Moreover, the main difference between our approach and MCBR is again that they focus on distributed retrieval.

Concerning team formation. Gomez, Abasolo and plaza [14] present a framework that allows teams of agents with different capabilities join together to solve a given problem. However, the main difference between the team formation literature and our work is that in team formation each agent is supposed to have only a subset of the capabilities required to solve a problem, and teams have to be formed to solve problems. In our approach, each agent is supposed to be able to generate a prediction for a given problem, and collaboration is only used to increase the accuracy with which predictions are made.

Moreover, team formation is sometimes called *Cooperative Problem Solving* (CPS) [8, 32]. Four stages are clearly identified in CPS: 1) potential recognition (finding which agents can perform certain tasks), 2) team formation, 3) plan formation, and 4) plan execution. This framework is certainly a general way to deal with team formation, however it focuses on finding a plan (or protocol) that the individual agents can follow to collaborative solve a given problem by combining their capabilities, and usually assumes that if two different agents are able to perform a task, it does not matter which of both is selected. In our framework, we deal with a more specific form of collaboration (committees), where all the agents are capable of predicting solutions,

but the selection of the specific members of the committee is crucial for the good performance of the committee.

Also relevant is work on learning to form coalitions of agents by Sarathi and Sen [9], where they propose a framework for agents that learn who are the best agents to collaborate with in the form of stable coalitions. However, they focus on the assignment of tasks to individual agents that can perform them in a more efficient way, rather than aggregating individual predictions as we do.

8. Conclusions and Future Work

We have presented a framework for collaborative multi-agent CBR systems called $\mathcal{M}$AC. The framework is collaborative in the sense that the agents collaborate with other agents if this can report some improvement in performance. This article addresses two main issues on collaboration: when to collaborate, and with whom to collaborate. We have presented the idea of committees to study these issues, and specifically presented a collaboration strategy called PB-CCS that allows the agents to learn when to convene committees, and which agents to invite to each committee. We have also presented a learning technique that allows an agent to learn its individual competence models, that are required by PB-CCS.

From the empirical evaluation we can conclude several things: first, PB-CCS is more robust than both CCS and B-CCS, since it achieves higher classification accuracy values in a wider range of scenarios than CCS or B-CCS; thus, we can say that the learnt competence models are more robust than the predefined ones used in B-CCS. Second, PB-CCS convenes in average smaller committees than CCS while achieving same accuracy (or higher, as in the untruthful agents scenario). And third, the proactive learning process acquires adequate competence models since PB-CCS behaves as expected in all the three scenarios. Moreover, given the experimental results, we can say that PB-CCS will perform well (i.e. having a high accuracy) if a) the agents have a reasonable number of cases (needed to collect M-examples), b) the agents do not change their behavior radically (otherwise the competence models wouldn't predict well their behavior), and c) there are at least some competent and truthful agents in the system (otherwise no collaboration strategy can perform well).

As future work, we plan to perform incremental learning, where the competence models should be updated as time passes. In this scenario, the competence models should be able to adapt if more agents enter or leave in the $\mathcal{M}$AC system, and to reflect changes in the kind of

problems that the system is solving. If the agents store the SERs from the other agents received when playing the role of the convener agents, the competence models could be updated by learning new trees reflecting the changes in the behaviors of the other agents. To detect when a competence model has to be updated, an agent could compare the behavior of an external agent with the predicted behavior from the learned competence model for that agent. When the learned competence model does not predict well the behavior of an external agent anymore, it has to be updated.

Also as future work, we plan to expand the scope of problems that the individual agents solve. We have presented results for classification tasks, but we plan to work with regression, planning and configuration tasks. To deal with other tasks, new aggregation methods for the individual predictions has to be designed, since voting would not work (for regression domains, weighted averaging could be used). Moreover, other tasks will need quality measures of the prediction of agents in order to evaluate when an agent has correctly solved a problem, and thus be able to learn competence models. Finally, notice that the $\mathcal{M}$AC framework is general enough to be applicable to any other task than classification given that: a) aggregation methods for predictions can be defined, and b) the correctness of a prediction can be assessed.

Acknowledgements

The authors thank Josep-Lluís Arcos of the IIIA-CSIC for their support and for the development of the Noos agent platform. Support for this work came from projects TIC2000-1414 "eInstitutor" and (MCYT-FEDER) TIC2002-04146-C05-01 "SAMAP".

References

1. Aamodt, A. and E. Plaza: 1994, 'Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches'. *Artificial Intelligence Communications* **7**(1), 39–59. online at <url:http://www.iiia.csic.es/People/enric/AICom_ToC.html>.
2. Aha, D. (ed.): 1997, *Lazy Learning*. Kluwer Academic Publishers.
3. Brams, S. J. and P. C. Fishburn: 1983, *Approval Voting*. Birkhauser, Boston.
4. Breiman, L.: 1996, 'Bagging Predictors'. *Machine Learning* **24**(2), 123–140.
5. Cestnik, B. and I. Bratko: 1991, 'On estimating probabilities in tree pruning'. In: *Machine Learning-European Working Session on Learning-91*, Vol. 482 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, pp. 151–163.
6. Chan, P. K. and S. J. Stolfo: 1995, 'A comparative evaluation of voting and meta-learning on partitioned data'. In: *Proc. 12th Int. Conf. on Machine Learning*. pp. 90–98.
7. Cover, T. and P. Hart: 1967, 'Nearest neighbor pattern classification'. *IEEE Transactions on Information Theory* **13**(1), 21–27.

8. Dignum, F., B. Dunin-Kęplicz, and R. Verbrugge: 2001, 'Agent Theory for Team Formation by Dialogue'. *Lecture Notes in Computer Science* **1986**, 150–??
9. Dutta, P. S. and S. Sen: 2002, 'Emergence of Stable Coalitions via Task Exchanges'. In: C. Castelfranchi and W. L. Johnson (eds.): *Proc. 1st Int. Conf. on Automous Agents and Multiagent Systems*. pp. 312–313.
10. Esteva, M., J. Padget, and C. Sierra: To appear, 'Formalising a language for institutions and norms'. In: *Intelligent Agents VIII, Proceedings ATAL'01*.
11. Esteva, M., J. A. Rodriguez-Aguilar, C. Sierra, P.Garcia, and J. L. Arcos: 2001, 'On the formal specification of electronic institutions'. In: *Agent Mediated Electronic Commerce*, Vol. 1991 of *LNAI*. Springer-Verlag.
12. Freund, Y. and R. E. Schapire: 1996, 'Experiments with a new Boosting algorithm'. In: *Proc. 13th Int. Conf. on Machine Learning*. pp. 148–146.
13. Gama, J.: 1998, 'Local cascade generalization'. In: *Proc. 15th Int. Conf. on Machine Learning*. pp. 206–214.
14. Gomez, M., C. Abasolo, and E. Plaza: 2001, 'Domain-Independent Ontologies for Cooperative Information Agents'. Vol. 2182 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, pp. 118–129.
15. Hansen, L. K. and P. Salamon: 1990, 'Neural networks ensembles'. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (12), 993–1001.
16. Leake, D. B. and R. Sooriamurthi: 2001, 'When Two Case Bases Are Better than One: Exploiting Multiple Case Bases'. In: *ICCBR*. pp. 321–335.
17. Leake, D. B. and R. Sooriamurthi: 2002, 'Managing Multiple Case Bases: Dimensions and Issues'. In: *Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society (FLAIRS)*. pp. 106–110.
18. McGinty, L. and B. Smyth: 2001, 'Collaborative Case-Based Reasoning: Applications in Personalized Route Planning'. In: *Case Based Reasoning ICCBR-01*. pp. 362–376.
19. Ontañón, S.: 2005, 'Ensemble Case Based Learning for Multi-Agent Systems'. Ph.D. thesis, Universitat Autònoma de Barcelona.
20. Ontañón, S. and E. Plaza: 2002a, 'A bartering aproach to improve multiagent learning'. In: *1st Int. Joint Conference in Autonomous Agents and Multiagent Systems*.
21. Ontañón, S. and E. Plaza: 2002b, 'Collaboration Strategies to Improve Multiagent Learning'. *Lecture Notes in Artificial Intelligence* **2430**, 331–344.
22. Ontañón, S. and E. Plaza: 2003a, 'Collaborative Case Retention Strategies for CBR Agents.'. *Lecture Notes in Artificial Intelligence* **2689**, 392–406.
23. Ontañón, S. and E. Plaza: 2003b, 'Justification-based Multiagent Learning,'. In: *International Conference on Machine Learning ICML-2003*. pp. 576–583.
24. Ontañón, S. and E. Plaza: 2004, 'Justification-based Case Retention'. In: *European. Conf. Case Based Reasoning (ECCBR 2004)*. pp. 346–360.
25. Ontañón, S. and E. Plaza: 2005, 'Recycling Data for Multi-Agent Learning'. In: *Proc. 22nd International Conference on Machine Learning ICML-2005*. pp. 633–640.
26. Ontañón, S. and E. Plaza: 2006, 'Arguments and Counterexamples in Case-based Joint Deliberation.'. In: *Workshop on Argumentation on Multi-Agent Systems (2006)*. p. to appear.
27. Perrone, M. P. and L. N. Cooper: 1993, 'When networks disagree: Ensemble methods for hybrid neural networks'. In: *Artificial Neural Networks for Speech and Vision*. Chapman-Hall.
28. Plaza, E. and S. Ontañón: 2001, 'Ensemble Case-based Reasoning: Collaboration Policies for Multiagent Cooperative CBR'. In: I. Watson and Q. Yang

(eds.): *In Case-Based Reasoning Research and Development: ICCBR-2001*. pp. 437–451.
29. Plaza, E. and S. Ontañón: 2003, 'Cooperative Multiagent Learning'. *Lecture Notes in Artificial Intelligence* **2636**, 1–17.
30. Quinlan, J. R.: 1986, 'Induction of Decision Trees'. *Machine Learning* **1**(1), 81–106.
31. Wolpert, D. H.: 1990, 'Stacked Generalization'. Technical Report LA-UR-90-3460, Los Alamos, NM.
32. Wooldridge, M. and N. R. Jennings: 1994, 'Towards a Theory of Cooperative Problem Solving'. In: *Proc. Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-94)*. Odense, Denmark, pp. 15–26.

Address for Offprints: Santi Ontañón, Artificial Intelligence Research Institute (IIIA), Consejo Superior de Investigaciones Científicas (CSIC), Campus UAB, 08193, Bellaterra, Catalonia, Spain